

INTEGRATED PLANT DISEASE DETECTION SYSTEM

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

TEJASWI. R (40111322)

SRINIVAS. R (40111269)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category-1 University by UGC

Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of TEJASWI. R (Reg. No - 40111322), SRINIVAS. R (Reg. No - 40111269) who carried out the Project Report entitled "INTEGRATED PLANT DISEASE DETECTION SYSTEM" under my supervision from November 2023 to April 2024.

Internal Guide

R. Aishwarya M.E., (Ph.D.,)

Head of the Department

Dr. L. LAKSHMANAN M.E., Ph.D.,

HEAD OF THE DEPARTMENT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Submitted for Viva voce Examination on 12/04/24

Chennai - 600 119.

Internal Examiner

External Examiner

DECLARATION

I, **TEJASWI. R** (Reg. No 40111322), hereby declare that the Project Report entitled "**INTEGRATED PLANT DISEASE DETECTION SYSTEM**" done by me under the guidance of **R. Aishwarya M.E., (Ph.D.,)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 12 - 04 - 24

PLACE: Chennai

R. Tejaswi

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **R. Aishwarya M.E., (Ph.D.,)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Nowadays crops are affected by different types of diseases which farmers cannot find what disease it is. It takes more time to know about the affected disease and finding cure for the disease is a long time process. Within that time the crop will be affected by the disease severely or it may die. To make this disease detection process more easier we have introduced our web application or mobile application which makes the disease detection process more easier and it takes less amount of time. This can help farmers as well as people to easily detect which type of disease is affected to the crop and suggest the best ways to overcome the issues. This process is cost efficient when compared to other techniques. This system introduces an Automated Plant Disease Identification System using hybrid Deep Learning model, which involves merging of Xception and ResNet50 architectures. Focused on apples, corn, and tomatoes, the system aids farmers in prompt and precise disease diagnosis. This also helps people to do agriculture in more efficient and easy ways in small area. Leveraging diverse datasets, the model classifies diseases such as apple, corn and tomato. Stochastic gradient descent optimizes model performance, facilitating practical deployment through a user-friendly web application. The system aims to enhance agriculture efficiency, contribute to sustainable practice, and ensure food security by addressing the challenges posed by plant diseases. The model achieved an accuracy of 0.9962 with a considerable amount of loss 0.0155.

KEYWORDS:

Automated Plant Disease Identification, Hybrid Deep Learning, Xception, ResNet50, Agricultural Management, Crop Diseases, Stochastic Gradient Descent, User-Friendly Web Application, Sustainable Agriculture, Food Security.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF FIGURES	VII
	LIST OF ABBREVIATIONS	VIII
1	INTRODUCTION	01
2	LITERATURE REVIEW	03
3	AIM AND SCOPE	05
	3.1 Aim	05
	3.2 Scope	06
	3.3 Hardware requirements	07
	3.4 Software used	08
4	DESCRIPTION OF PROPOSED SYSTEM	11
	4.1 Methodology	11
	4.2 Methods And Techniques Used	13
	4.3 Algorithms Used	15
	4.4 Existing System and disadvantages	16
	4.5 Proposed System and advantages	17
	4.6 Architecture Diagram of proposed system	18
	4.7 Flowchart	19
5	RESULT AND DISCUSSION	20
6	SUMMARY AND CONCLUSION	26
	REFERENCES	28
	APPENDIX	30
	A. Source Code	30
	B. Screenshots	42
	C. Research Paper	51
	D. Certificate	57

LIST OF FIGURES

Fig. No.	Figure Name	Page No.
4.8	Architecture Diagram of Proposed system	18
4.9	Flow Chart	19
5.1	Accuracy of the model	20
5.2	Loss of the model	21
5.3	Confusion Matrix	21
5.4	Precision Values	22
5.5	Apple Cedar Apple Rust Disease Datasets	22
5.6	Apple Black Rot Disease Datasets	23
5.7	Corn Common Rust Disease Datasets	23
5.8	Corn Northern Leaf Blight Disease Datasets	24
5.9	Tomato Mosaic Virus Disease Datasets	24
5.10	Tomato Yellow Leaf Curl Virus Disease Datasets	25
B.1	Importing necessary packages	42
B.2	Assigning size, training and testing datasets	42
B.3	Initialize training methods and training rate	43
B.4	Model Accuracy	43
B.5	Accuracy Analysis	44
B.6	Accuracy and loss of the model	44
B.8	Splitting and resize	45
B.9	CN Matrix	45
B.10	Confusion Matrix analysis	46
B.11	Precision Values	46
B.12	Front page of Web application	47
B.13	Uploading dataset and fetching outputs	47

LIST OF ABBREVIATIONS

ABBREVIATIONS	FULL FORM
ML	Machine Learning
DL	Deep Learning
Xception	Extreme Inception
ResNet50	Residual Network
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
SGD	Stochastic Gradient Descent

CHAPTER 1

INTRODUCTION

Plant diseases pose a significant threat to global agriculture, jeopardizing crop yields, food security, and economic stability. Addressing this challenge requires timely and accurate identification of diseases to enable effective crop management and disease mitigation strategies. However, traditional methods of disease diagnosis in agriculture often rely on subjective visual inspection by farmers, which can be error-prone and inefficient. In response to these limitations, this project proposes the development of an Automated Plant Disease Identification System. Leveraging the power of machine learning, specifically through a hybrid deep learning model combining Xception and ResNet50 architectures, this system aims to revolutionize the way plant diseases are identified and managed in agriculture. The significance of addressing plant diseases cannot be overstated. With the global population steadily increasing, the demand for food is escalating, placing immense pressure on agricultural systems to produce more crops. However, the spread of plant diseases threatens to undermine these efforts by reducing crop yields and compromising food security. Moreover, plant diseases have far-reaching economic implications, affecting the livelihoods of millions of farmers worldwide and impacting the stability of agricultural markets. By addressing plant diseases effectively, we can safeguard agricultural productivity, ensure food security, and promote economic stability. Traditional methods of disease diagnosis in agriculture, such as visual inspection by farmers, suffer from several limitations. Firstly, visual inspection is subjective and relies heavily on the expertise and experience of the observer, making it prone to errors and misdiagnosis. Additionally, manual inspection can be time-consuming and labor-intensive, particularly for large-scale agricultural operations. Furthermore, the effectiveness of manual methods may be limited by factors such as environmental conditions, the stage of disease progression, and the diversity of plant pathogens. As a result, there is a pressing need for automated solutions that can provide rapid and accurate identification of plant diseases.

Introducing an Automated Plant Disease Identification System has the potential to revolutionize the way plant diseases are managed in agriculture. By harnessing advanced technologies such as machine learning, this system can provide farmers and agricultural professionals with a reliable and efficient tool for the rapid diagnosis of plant diseases. Automated disease identification systems offer several advantages over traditional methods. They can analyze large volumes of data quickly and accurately, enabling early detection of diseases and timely intervention measures. Moreover, these systems can reduce reliance on subjective human judgment, leading to more consistent and reliable diagnoses. Overall, the introduction of an automated system has the potential to significantly enhance agricultural productivity, improve food security, and contribute to sustainable agriculture practices. The objectives of this project include developing a robust Automated Plant Disease Identification System capable of accurately identifying a wide range of plant diseases. The scope of the project encompasses the development of a hybrid deep learning model integrating Xception and ResNet50 architectures, as well as the implementation of a user-friendly interface for farmers and agricultural professionals. The project will be structured into several phases, including data collection and preprocessing, model development and training, system integration, and evaluation. Throughout each phase, emphasis will be placed on ensuring the reliability, accuracy, and usability of the automated system. By leveraging technology to revolutionize plant disease identification, this project aims to empower farmers with the tools they need to protect their crops, enhance agricultural productivity, and promote sustainable agriculture practices.

CHAPTER 2

LITERATURE REVIEW

Muhammad Hammad Saleem et.al.[1] implemented a deep learning model to identify disease affected the crops by comparing all the available techniques and found the best technique for their work. They used the best technique called Xception which accrued high accuracy and it was trained to identify 26 diseases in 14 different plants. They used this deep learning technique to identify plant disease which then resulted in accuracy of 99.8%.

Janani Varthini et.al.[2] suggested a deep learning model to identify the disease in the cotton plant combining two effective CNN architectures called Xception and ResNet50 by using a large amount of dataset so that farmers will find it easier to identify disease in cotton plant. They acquired an accuracy of 98%.

Vandana B. Malode et.al.[3] introduced a deep learning model to identify the disease infected the crops especially tomato and grapes using the Convolutional Neural Network (CNN) method which is used for detecting Multiple Crops Leaf Disease (MCLD). The CNN based architectural network known as Visual Geometry Group (VSG) was used by them. They used these techniques to show precise and accurate results for the crops and accrued an accuracy of 98.40%.

G. Sumalatha et.al.[4] presented a deep learning model to identify diseases in plants using the Convolutional Neural Network (CNN) and Deep Neural Network (DNN). Their goal was to create a system which can identify the disease using a simple leaf image. They used six different CNN architectures to implement the system from which DenseNet21 Architecture acquired an higher accuracy of 95.48% by detecting eight types of diseases.

Usman Abdur Rehman et.al.[5] using Deep Learning techniques he proposed a system called ResNet50 to identify diseases in corn, potatoes and Tomatoes. The ResNet50 performance was very high when compared with other two models, VGG16 and MCNN, which was built and also was trained from the beginning. ResNet50 gave the best results, achieving 98.7% accuracy in predicting the plant disease.

Sakshi Raina et. al. [6] Presented a deep learning Technique model to detect plant leaf disease using the images of the leaf. They used artificial intelligence and machine learning to analyze plant leaf images. They performed various key steps like image acquisition, image preprocessing etc. based on classifier to identify the disease. They proposed a system to detect plant disease quickly and accurately from pictures.

J. S. H. Al-bayati and B. B. Ustundag & et al. [7] Apple leaf disease is the foremost factor that restricts apple yield and quality. Usually, much time is taken for disease detection with the existing diagnostic techniques; therefore, farmers frequently miss the best time for preventing and treating diseases. The detection of apple leaf diseases is a significant research problem, and its main aim is to discover an efficient technique for disease leaf image diagnosis. This article has made an effort to propose a method that can detect the disease of apple plant leaf using deep neural network (DNN). Plant diseases detection system (PDDS) architecture is designed. Speeded up robust feature (SURF) is used for feature extraction and Grasshopper Optimization Algorithm (GOA) for feature optimization, which helps to achieve better detection and classification accuracy. Classification parameters, such as Precision, Recall, F-measure, Error, and Accuracy is computed, and a comparative analysis has been performed to depict the effectiveness of the proposed work.

CHAPTER 3

AIM AND SCOPE

3.1 AIM

The primary aim of this project is to develop an advanced and automated system for the identification of plant diseases affecting apples, corn, and tomatoes. Leveraging state-of-the-art deep learning techniques, the system targets the precise and rapid classification of various plant diseases, providing farmers and agricultural professionals with an efficient tool for early detection and intervention. The overarching goal is to enhance agricultural productivity and sustainability by significantly reducing the reliance on manual inspection methods and facilitating timely and accurate disease diagnosis. Through the integration of machine learning models and user-friendly interfaces, the system aims to empower users with accessible and actionable insights, contributing to improved crop management and, consequently, the overall well-being of the agricultural ecosystem.

Plant diseases present a formidable threat to global agriculture, impacting crop yields, food security, and economic stability. The timely and accurate identification of these diseases is crucial for effective crop management and disease mitigation. Traditional methods of disease diagnosis in agriculture often rely on visual inspection by farmers, which can be subjective and prone to errors. In response to these challenges, this project proposes an Automated Plant Disease Identification System that harnesses the power of machine learning, specifically a hybrid deep learning model amalgamating Xception and ResNet50 architectures.

We will discuss the significance of addressing plant diseases, the limitations of current manual methods, and the potential impact of introducing an automated system. Additionally, we will provide an overview of the objectives, scope, and structure of the project, emphasizing the role of technology in revolutionizing plant disease identification and contributing to sustainable agriculture.

3.2 SCOPE:

1. **Development of a Hybrid Deep Learning Model:** The project involves the development and implementation of a hybrid deep learning model that combines the strengths of Xception and ResNet50 architectures. This model will be trained on diverse datasets to accurately identify diseases affecting apples, corn, and tomatoes.
2. **User-Friendly Web Application:** A user-friendly web application will be developed to provide farmers and agricultural professionals with an intuitive interface for uploading plant images and receiving rapid disease diagnosis results. The application will leverage the trained deep learning model for efficient disease identification.
3. **Enhanced Agricultural Efficiency:** The proposed system aims to improve agricultural efficiency by enabling timely and accurate disease diagnosis, leading to effective disease management strategies. This will help minimize crop losses, optimize resource utilization, and contribute to increased crop yields.
4. **Contribution to Sustainable Agriculture:** By leveraging advanced technologies like deep learning, the project seeks to contribute to sustainable agricultural practices. The automated disease identification system will reduce reliance on manual inspection methods, minimize pesticide usage, and promote environmentally-friendly farming practices.
5. **Empowerment of Farmers:** The system will empower farmers with accessible and actionable insights, enabling them to make informed decisions about disease management and crop protection. By providing timely interventions, the project aims to enhance farmers' livelihoods and ensure food security.

6. **Continuous Improvement and Adaptability:** The project will emphasize continuous improvement and adaptability to evolving challenges and emerging diseases. Feedback from users and collaboration with agricultural experts will be integrated to refine the system and ensure its effectiveness in real-world agricultural settings.

Overall, the aim of the project is to develop an innovative solution for automated plant disease identification, with a scope encompassing technological innovation, agricultural efficiency, sustainability, and empowerment of farming communities.

3.3 REQUIREMENT ANALYSIS

3.3.1 Hardware Requirements:

1. **Processor (CPU):** Multi-core processor (e.g., quad-core or higher) for accelerated training times.
2. **Graphics Processing Unit (GPU):** Dedicated GPU, preferably from NVIDIA (e.g., GeForce GTX or RTX series) with CUDA support for faster deep learning model training.
3. **Memory (RAM):** Minimum 16 GB RAM, with larger sizes recommended for extensive datasets or concurrent tasks.
4. **Storage:** SSD storage for faster data access and model loading, with sufficient capacity based on dataset and model sizes.
5. **Networking:** Stable and high-speed internet connection for dataset downloads, model updates, or cloud-based collaboration.
6. **Additional Considerations:** Ensure hardware compatibility with deep learning frameworks, libraries, and consider cloud-based solutions for scalability if local resources are insufficient.

3.3.2 Software Used :

1. **Deep Learning Frameworks:**

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications. TensorFlow is commonly used for implementing and training deep learning models.

2. **Python Programming Language:** Python is widely utilized for machine learning and web development, serving as the primary language for coding the system. Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

Python Features :

- **Easy-to-learn** - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- **Easy-to-read** - Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** - Python's source code is relatively simple to maintain.
- **A broad standard library** - The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode** - Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- **Portable** - Python can operate on a broad range of hardware devices and has the same interface across them all.
- **Extendable** - The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- **Databases** - Python provides interfaces to all major commercial databases.
- **GUI Programming** - Python can construct and port GUI programs to numerous system calls, libraries, and windows systems, including Windows MFC, Macintosh, and Unix's X Window system.
- **Scalable** - Python provides a better structure and support for large programs than shell scripting.

3. **Keras:**

Keras is an API designed for human beings, not machines. Keras follows best practices for *reducing cognitive load*: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. Keras also gives the highest priority to crafting great documentation and developer guides. Often used in conjunction with TensorFlow, Keras provides a high-level neural networks API that simplifies the development process.

Advantages of Keras :

Keras has several advantages that make it a popular choice for deep learning.

- **Simplicity** - Keras is very easy and simple to use. It is a user-friendly API with easy-to-learn and code features.
- **Backend support** - Keras does not operate with low-level computations. So, it supports the use of backends.

- **Pre-trained models** - Keras provides numerous pre-trained models.
- **Fast experimentation** - Keras is built to simplify the tasks of users.
- **Great community and calibre documentation** - Keras has a large supportive community.

4. **Streamlit:** For building the user-friendly web application, Streamlit is a popular Python library that simplifies the creation of interactive and data-driven web apps.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 Methodology

Data Collection and Preparation

This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get, the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions and etc..

Careful compilation of a diverse dataset featuring labelled images showcasing various stages of plant diseases affecting apples, corn, and tomatoes.

Application of meticulous preprocessing techniques to standardize images, ensuring uniformity in resolution, color, and quality.

Dataset :

The dataset consists of 21,000 image datasets. They are separated into training dataset and validation dataset . The model is trained by separating the datasets into 6 labels.

Model Architecture Selection and Fusion

Selection of appropriate deep learning architectures post data preparation, considering the unique characteristics of diseases in each crop.

Fusion of selected architectures into a hybrid model to leverage the individual strengths for recognizing complex disease patterns.

Implementation using the most familiar DL frameworks like TensorFlow and PyTorch for efficient computation and flexibility for the training of the model.

Training and Optimization

Training of the hybrid DL model using the available dataset is performed. Learning rate, batch size, and optimizer setting parameters are fine tuned to acquire higher model performance.

Usage of techniques such as the stochastic gradient descent and methods of regularization are performed to prevent the overfitting and enhance the generalization of the model.

Iterative training-validation are performed to monitor the convergence and to upgrade the performance metrics.

Performance Evaluation and Validation

The Validation phase is used to assess the performance of the model for its unseen data segments.

Computation of evaluation metrics such as accuracy, precision, recall, and F1-score are performed to quantify the proficiency of the model.

Usage of the Cross-validation techniques and also the confusion matrix analysis to calculate the robustness of the model and also to identify the weakness of the model.

Integration and Deployment

Integration of the model into a web application that is user friendly using Streamlit upon achieving satisfactory performance.

Development of an intuitive interface enabling farmers and agricultural professionals to seamlessly upload images of plant samples for rapid and accurate disease predictions.

Strong testing is done to ensure the stability and compatibility of the application across variety of platforms.

Incorporation of feedback from users to refine the usability of the application, ensuring seamless integration into agricultural practices.

4.2 Methods and Technologies Used

Deep Learning Architectures:

Utilization of deep learning architectures, such as Xception and ResNet50, for effective feature extraction and pattern recognition in plant images.

Transfer Learning :

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

Implementation of the familiar transfer learning techniques to leverage the already trained models (Xception and ResNet50) for the efficient training of the hybrid model.

Optimization Techniques:

Fine tuning of the hybrid DL model using the available dataset is performed. Learning rate, batch size, and optimizer setting parameters are fine tuned to acquire higher model performance.

Stochastic Gradient Descent:

Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent algorithm that is used for optimizing machine learning models. It addresses the computational inefficiency of traditional Gradient Descent methods when dealing with large datasets in machine learning projects.

In SGD, instead of using the entire dataset for each iteration, only a single random training example (or a small batch) is selected to calculate the gradient and update the model parameters. This random selection introduces randomness into the optimization process.

Use of stochastic gradient descent as an optimization algorithm to iteratively update model weights, preventing overfitting and enhancing generalization capabilities.

Regularization Methods:

Usage of the regularization techniques such as dropout or L2 regularization, to control the overfitting of the data and improve the ability of the model to generalize the unseen data.

Cross-Validation:

Cross-validation techniques that includes k-fold cross-validation, to evaluate model robustness and identify potential weaknesses.

Confusion Matrix Analysis:

Utilization of confusion matrix analysis to provide insights into model performance, identifying areas of strengths and weaknesses in disease classification.

Streamlit for Web Application Development:

Streamlit is an open-source Python library for creating web applications for data science and machine learning projects. It's designed to be used by data scientists and machine learning engineers who do not have extensive front-end development skills. It has a simple syntax allowing you to create interactive web apps with a few lines of code. Adoption of Streamlit, a Python library, for the development of an

intuitive and user-friendly web application for seamless model integration and user interaction.

4.3 ALGORITHMS USED :

Xception:

Xception stands for “extreme inception” it takes the principles of Inception to an extreme. In Inception, 1x1 convolutions were used to compress the original input, and from each of those input spaces we used different type of filters on each of the depth space. Xception just reverses this step. Instead, it first applies the filters on each of the depth map and then finally compresses the input space using 1X1 convolution by applying it across the depth. This method is almost identical to a depth-wise separable convolution.

ResNet50:

ResNet50 is a powerful image classification model that can be trained on large datasets and achieve state-of-the-art results. One of its key innovations is the use of residual connections, which allow the network to learn a set of residual functions that map the input to the desired output. These residual connections enable the network to learn much deeper architectures than was previously possible, without suffering from the problem of vanishing gradients. The architecture of ResNet50 is divided into four main parts: the convolutional layers, the identity block, the convolutional block, and the fully connected layers. The convolutional layers are responsible for extracting features from the input image, while the identity block and convolutional block are responsible for processing and transforming these features. Finally, the fully connected layers are used to make the final classification.

4.4 Existing System and Disadvantages :

The current manual inspection for plant disease diagnosis is subjective and labor Intensive. It introduces variability in results and may lead to delays in disease identification, impacting agricultural productivity.

Subjectivity in disease diagnosis due to reliance on manual inspection methods. Human errors, influenced by factors like fatigue, impact the accuracy and consistency of disease diagnoses. Resource-intensive nature and limited scalability, particularly challenging with large datasets or a high volume of plant samples. Efficiency hindered by the heavy dependence on the expertise of individual farmers or agricultural professionals. Lack of standardized protocols and criteria for disease diagnosis leading to variations in reporting and diagnostic criteria.

Disadvantages of the Existing Plant Disease Identification System:

Manual Inspection Introduces Subjectivity:

- Subjectivity in disease diagnosis due to reliance on manual inspection methods.

Human Errors and Variability Impact Accuracy:

- Human errors, influenced by factors like fatigue, impact the accuracy and consistency of disease diagnoses.

Limited Scalability and Resource Intensity:

- Resource-intensive nature and limited scalability, particularly challenging with large datasets or a high volume of plant samples.

Dependence on Expertise Hinders Efficiency:

- Efficiency hindered by the heavy dependence on the expertise of individual farmers or agricultural professionals.

Absence of Standardized Protocols:

- Lack of standardized protocols and criteria for disease diagnosis leading to variations in reporting and diagnostic criteria

4.5 Proposed System and its advantages :

The proposed automated system employs a hybrid deep learning model for efficient and accurate plant disease identification. It aims to enhance the speed, precision, and scalability of disease diagnosis, reducing dependency on manual inspection and fostering sustainable agricultural practices.

The main goal is to implement an advanced and automatic system for the identification of plant diseases affecting apples, corn, and tomatoes. Leveraging state-of-the-art deep learning techniques, the system targets the precise and rapid classification of various plant diseases, providing farmers and agricultural professionals with an efficient tool for early detection and intervention. The overarching goal is to enhance agricultural productivity and sustainability by significantly reducing the reliance on manual inspection methods and facilitating timely and accurate disease diagnosis. Through the integration of machine learning models and user-friendly interfaces, the system aims to empower users with accessible and actionable insights, affording to improvised crop management and, consequently, the overall well-being of the agricultural ecosystem.

Automation for efficient and consistent plant disease identification. Improvisation of accuracy with the familiar and advanced ML models. Rapid processing for accurate interventions in agriculture. Enhanced scalability for handling large datasets. Reducing the dependency on individual expertise, making it accessible to a broader user base. User-friendly interface to increase the accessibility among the farmers. Often improvement through collaboration with agricultural experts. Drastic Enhancement of agriculture productivity and sustainability.

Advantages of Proposed System:

- Automation for efficient and consistent plant disease identification.
- Improved accuracy with advanced machine learning models.
- Rapid processing for timely interventions in agriculture.
- Enhanced scalability for handling large datasets.

- Reduced dependency on individual expertise, making it accessible to a broader user base.
- User-friendly interface for increased accessibility among farmers.
- Continuous improvement through collaboration with agricultural experts.
- Overall enhancement of agricultural productivity and sustainability.

4.6 Architecture Diagram of Proposed System:

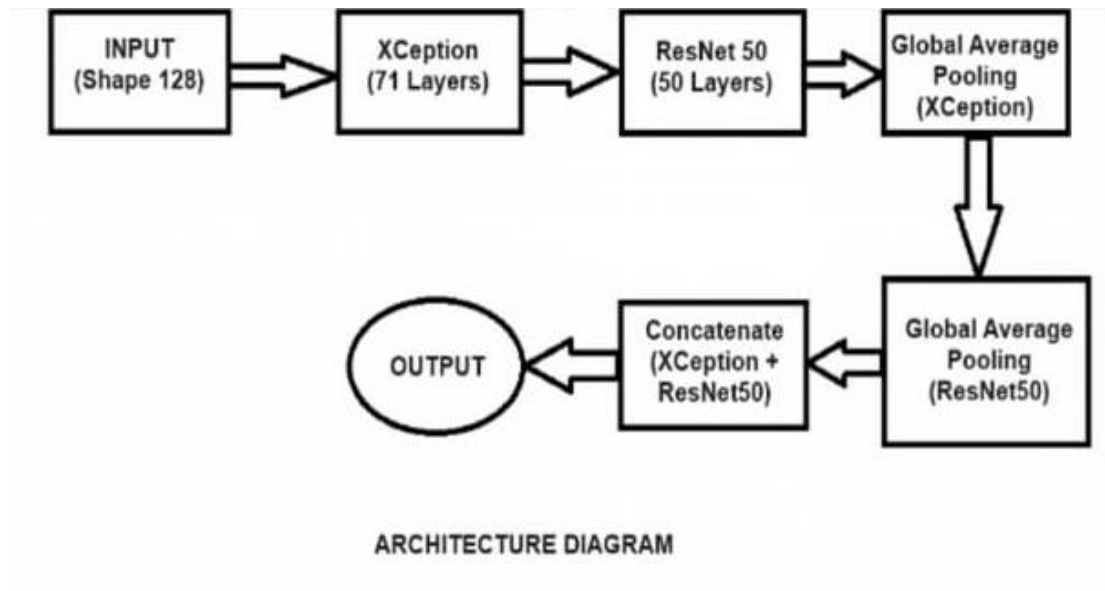


Fig 4.8 Architecture diagram of proposed system

4.7 Flowchart:

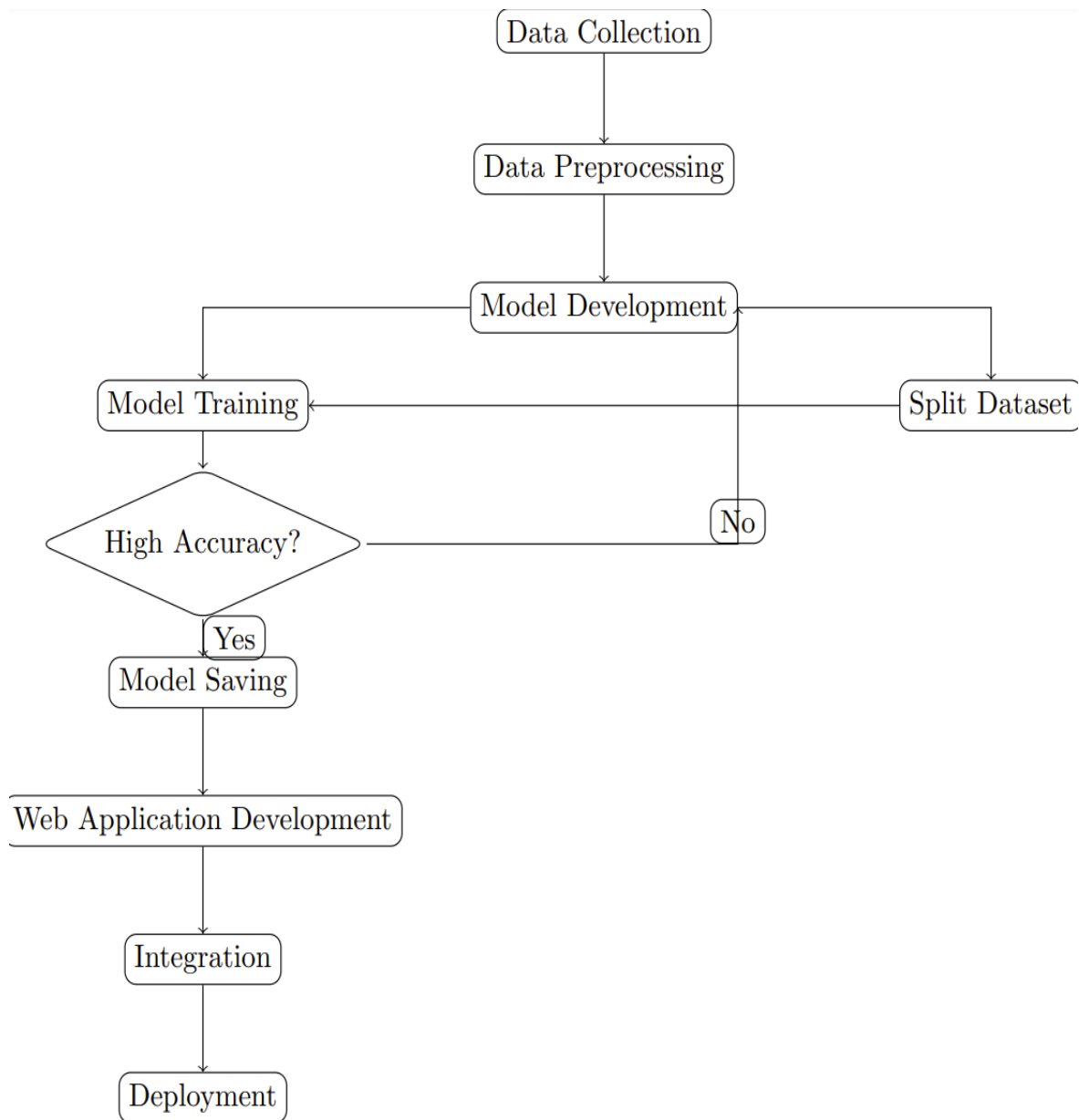


Fig 4.9 Flowchart

CHAPTER 5

RESULT AND DISCUSSION

The results obtained from the implementation of the proposed Automated Plant Disease Identification System using hybrid deep learning models is presented below. The performance metrics, including accuracy, loss, confusion matrix, and precision values are discussed. Additionally, sample datasets for each crop are provided to illustrate the effectiveness of the system.

The results will be based on the input given. The defected plant image is given as an input and our trained model identifies the disease affected the particular crop precisely. The model achieved an accuracy of 0.9962 with a considerable amount of loss 0.0155.

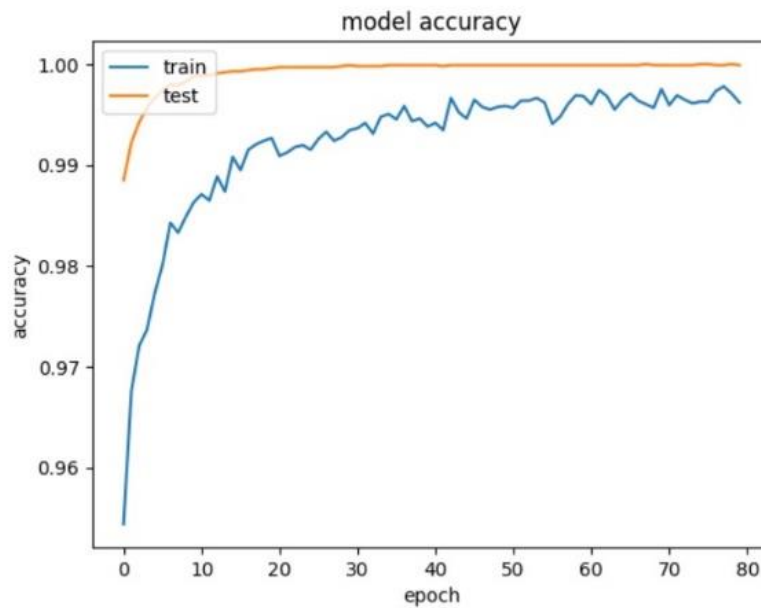


Fig 5.1 Accuracy of the model

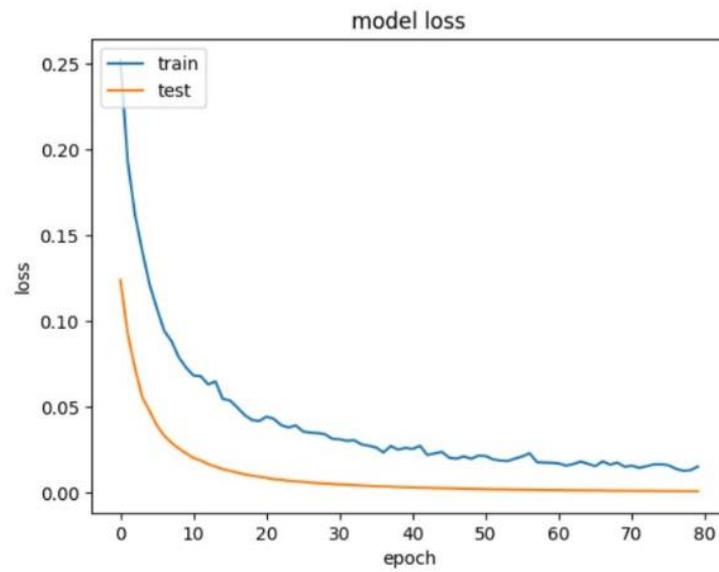


Fig 5.2 Loss of the model

Confusion matrix:

```
[[2484    0    0    0    0    0]
 [   0 2200    0    0    0    0]
 [   0    0 2384    0    0    0]
 [   0    0    0 2385    0    0]
 [   0    0    0    0 2451    0]
 [   0    0    0    0    0 2238]]
```

Accuracy: 1.0

Fig 5.3 Confusion Matrix

```

True Positives: [2484 2200 2384 2385 2451 2238]
True Negatives: 0
Specificity: 1.0
Precision: 1.0
Recall: 1.0

```

Fig 5.4 Precision Values

5.1 Sample Datasets :

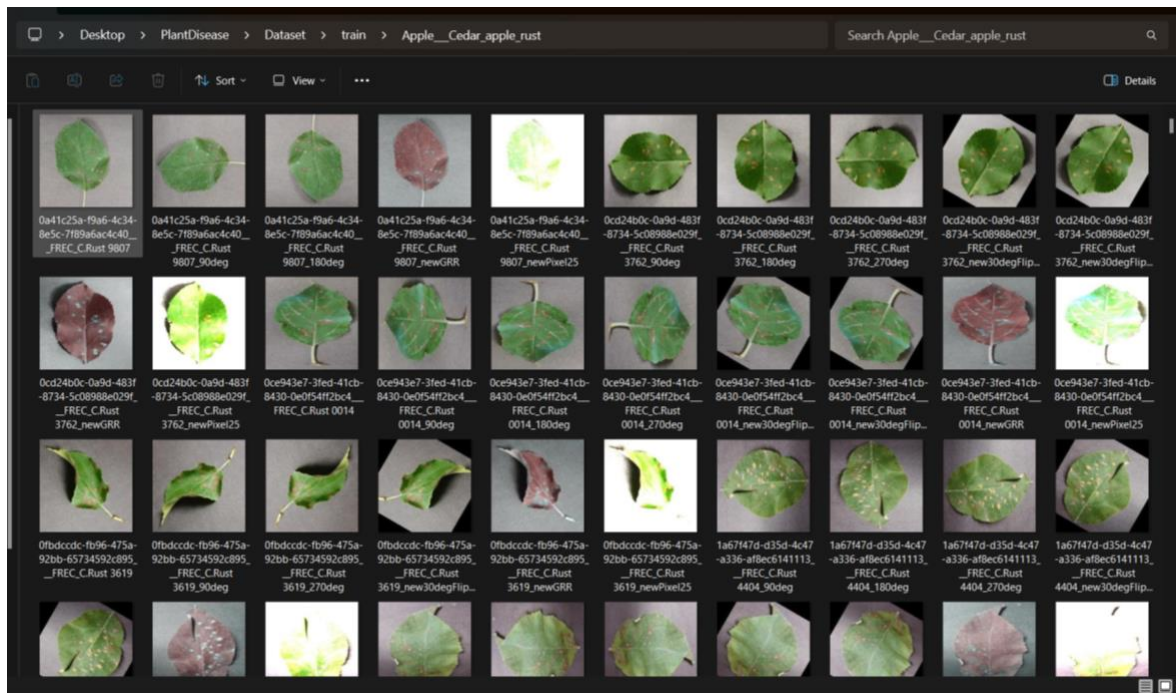


Fig 5.5 Apple Cedar Apple Rust Disease Datasets

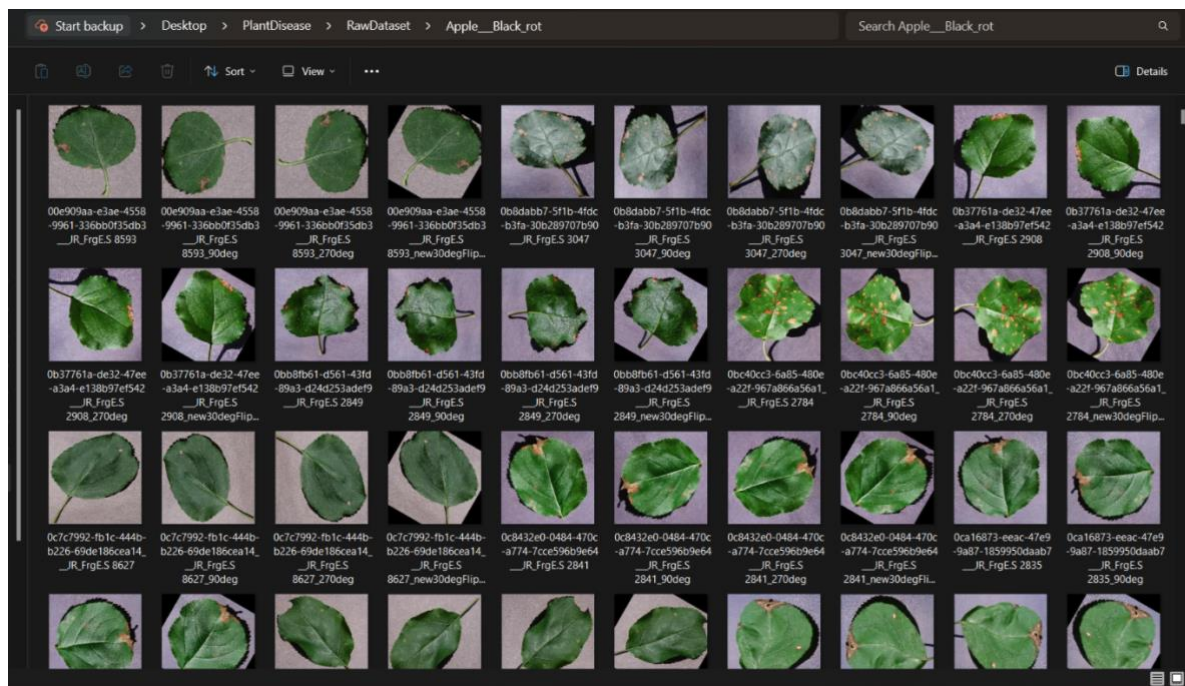


Fig 5.6 Apple Black Rot Disease Datasets

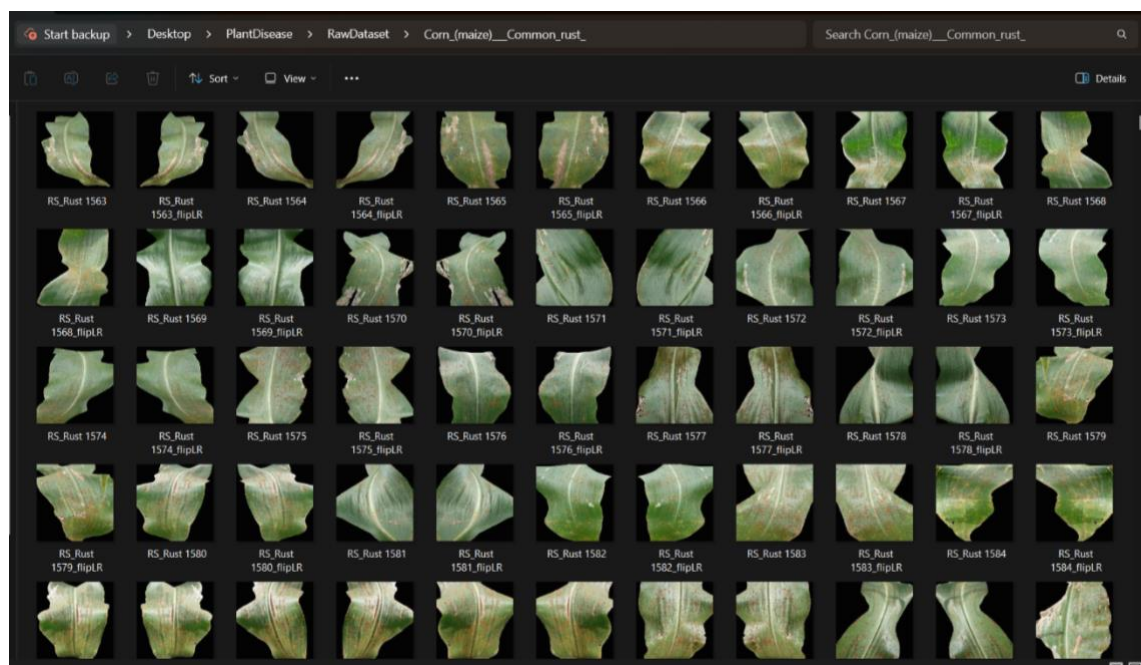


Fig 5.7 Corn Common Rust Disease Datasets

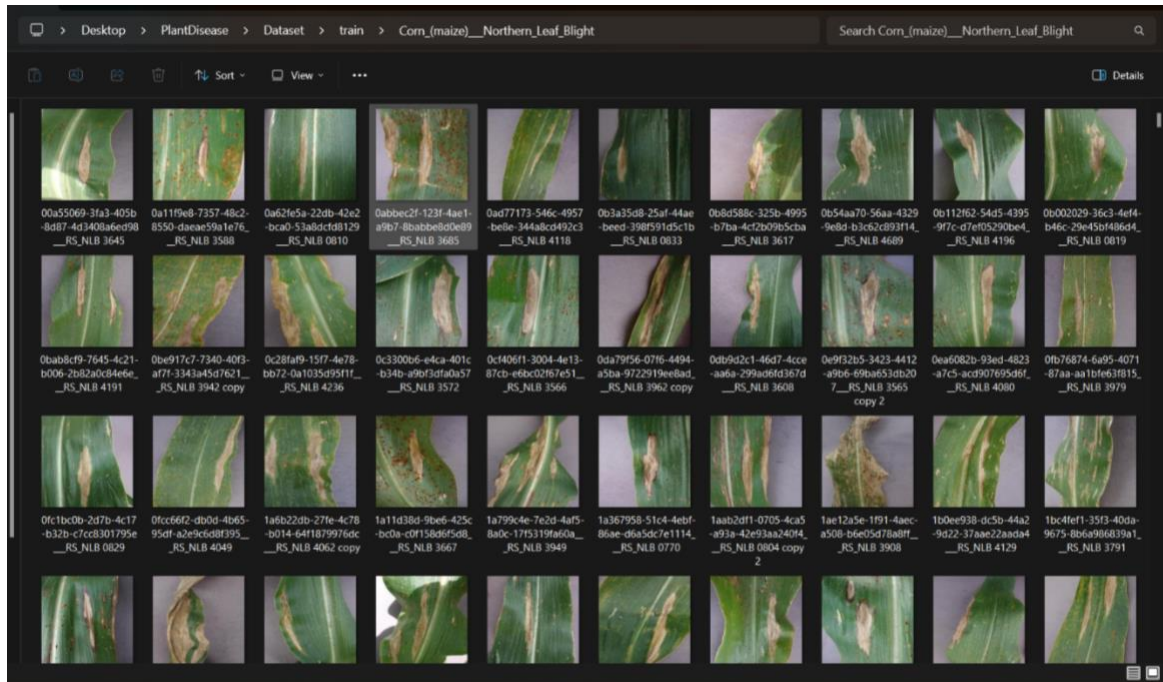


Fig 5.8 Corn Northern Leaf Blight disease datasets

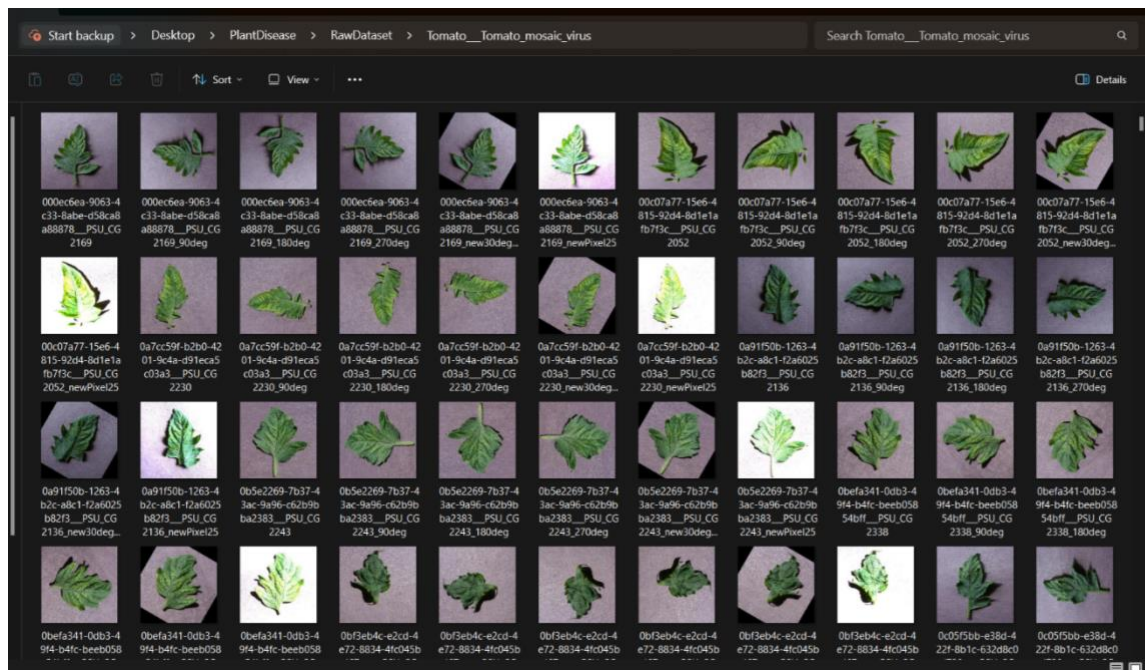


Fig 5.9 Tomato Mosaic Virus Disease Datasets

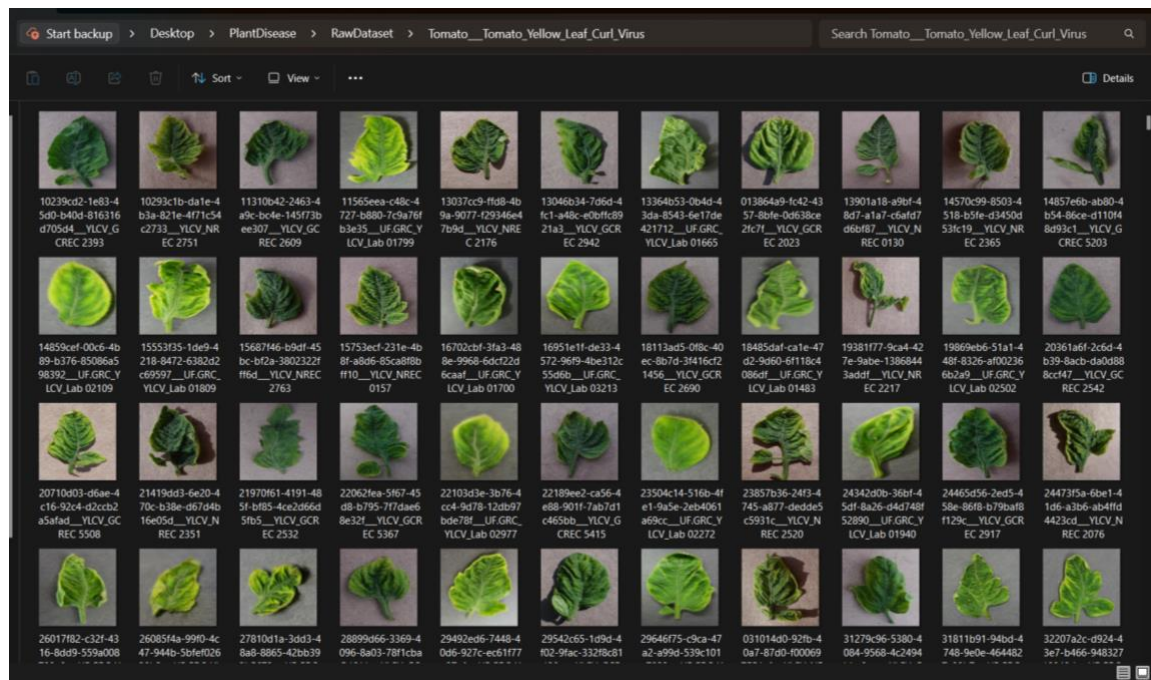


Fig 5.10 Tomato Yellow Leaf Curl Virus Disease Datasets

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 SUMMARY :

The proposed system provides a significant leap forward in the system of plant disease identification through the hybrid deep learning models to the agricultural practices. This systems advantages includes consistent identification and improved accuracy through increased scalability and also a sustainable approach to managing the diseases in plants. It reduces the dependency of manual inspection our system allows access to accurate disease diagnosis benefiting the farmers. This system ensures its adaptability to evolving challenges and emerging diseases for the selected crops. This system has the potential to significantly improvise the agricultural productivity by identifying the plant diseases in a more efficient and precise manner for the selected crops. The model achieved an accuracy of 0.99 and an considerable amount of loss rate 0.16.

6.2 CONCLUSION :

In conclusion, the development and implementation of an Automated Plant Disease Identification System using a hybrid Deep Learning model represent a significant advancement in agricultural technology. Through the fusion of Xception and ResNet50 architectures, our system offers a robust solution for prompt and accurate diagnosis of diseases affecting apples, corn, and tomatoes.

The project addresses the critical need for timely disease identification in agriculture, aiming to mitigate the adverse effects of plant diseases on crop yield, food security, and economic stability. By leveraging machine learning techniques, we have created a scalable and efficient tool that empowers farmers and agricultural professionals to make informed decisions and take proactive measures against crop diseases.

Our system's advantages include automation for consistent disease identification, improved accuracy through advanced machine learning models, and rapid processing for timely interventions in agriculture. Additionally, the system enhances scalability for handling large datasets and reduces dependency on individual expertise, making it accessible to a broader user base.

Through extensive research and development, we have demonstrated the efficacy of our system in accurately classifying various plant diseases, achieving high levels of accuracy and performance. The integration of user-friendly interfaces further enhances the accessibility and usability of the system, enabling seamless interaction for farmers and agricultural stakeholders.

Overall, the proposed system holds immense potential to revolutionize agricultural practices, enhancing productivity, sustainability, and food security. Continuous refinement and collaboration with agricultural experts will further improve the system's capabilities and ensure its adaptability to evolving challenges in plant disease management. By harnessing the power of technology, we can pave the way for a more resilient and efficient agricultural ecosystem, ultimately benefiting farmers, consumers, and the global economy.

REFERENCES

- [1] Muhammad Hammad Saleem, Johan Potgieter and Khalid Mahmood Arif "Plant Disease Classification: A Comparative Evaluation of Convolutional Neural Networks and Deep Learning Optimizers" 2020 by the authors. Licensee MDPI, Basel, Switzerland. doi:10.3390/plants9101319
- [2] Vani Rajasekar, K. Venu, Soumya Ranjan Jena, R. Janani Varthini and S. Ishwarya. 2021 Detection of Cotton Plant Diseases Using Deep Transfer Learning. Journal of Mobile Multimedia, Vol. 18 2, 307–324. doi: 10.13052/jmm1550-4646.1828 River Publishers.
- [3] Ananda S. Paymode , Vandana B. Malode. 2022 Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG. The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co., Ltd. This is an open access article under the CC BY-NC-ND license.
- [4] G.Sumalatha, Dr Jhansi Rani Singothu, Dr S. Krishna Rao. Transfer learning-based Plant Disease Detection.
- [5] Muhammad Sufyan Arshad, Usman Abdur Rehman, Muhammad Moazam Fraz. Plant Disease Identification Using Transfer Learning. 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2) | 978-1-6654-1285-8/20/\$31.00 ©2021 IEEE DOI:10.1109/ICoDT252288.2021.9441512.
- [6] Sakshi Raina, Dr. Abhishek Gupta. A Study on Various Techniques for Plant Leaf Disease Detection Using Leaf Image. 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) | 978-1-7281-9537-7/20/\$31.00 ©2021 IEEE DOI:10.1109/ICAIS50930.2021.9396023.
- [7] Rajbongshi, A., Khan, T., Pramanik, M. M. R. A., Tanvir, S. M., & Siddiquee, N. R. C. (2021). Recognition of mango leaf disease using convolutional neural network models: a transfer learning approach. Indonesian Journal of Electrical Engineering and Computer Science, 23(3), 1681-1688.
- [8] Verma, S., Chug, A., Singh, A. P., Sharma, S., & Rajvanshi, P. (2019). Deep learning-based mobile application for plant disease diagnosis: A proof of concept

with a case study on tomato plant. In Applications of image processing and soft computing systems in agriculture (pp. 242-271). IGI global.

[9] Aversano, L., Bernardi, M. L., Cimitile, M., Iammarino, M., & Rondinella, S. (2020, November). Tomato diseases classification based on VGG and transfer learning. In 2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor) (pp. 129-133). IEEE.

[10] ÖZDEN, C. (2023). Corn Disease Detection Using Transfer Learning. Black Sea Journal of Engineering and Science, 387-393.

[11] Fauzi, A., Syarif, I., & Badriyah, T. (2023). Development of a Mobile Application for Plant Disease Detection using Parameter Optimization Method in Convolutional Neural Networks Algorithm. *EMITTER International Journal of Engineering Technology*, 11(2), 192-213.

[12] Trihardianingsih, L., Sunyoto, A., & Hidayat, T. (2023). Classification of Tea Leaf Diseases Based on ResNet-50 and Inception V3. Sinkron: jurnal dan panellation Teknik informatika, 8(3), 1564-1573.

[13] Altuntaş, Y., & KOCAMAZ, F. (2021). Deep feature extraction for detection of tomato plant diseases and pests based on leaf images. *Celal Bayar University Journal of Science*, 17(2), 145-157.

[14] Ahmed, M. R. (2021). Leveraging convolutional neural network and transfer learning for cotton plant and leaf disease recognition. *Int. J. Image Graph. Signal Process*, 13(4), 47-62.

[15] Sharma, M., Kumar, C. J., & Deka, A. (2022). Early diagnosis of rice plant disease using machine learning techniques. *Archives of Phytopathology and Plant Protection*, 55(3), 259-283.

APPENDIX

A. SOURCE CODE

Importing necessary libraries:

```
import numpy as np

import warnings

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import tensorflow as tf

import keras

from keras.preprocessing.image import ImageDataGenerator

import keras.preprocessing.image

from tensorflow.keras.applications.resnet50 import ResNet50

from keras.applications.xception import Xception

from keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D,
Average, Input, Concatenate, GlobalMaxPooling2D, Conv2D

from keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.optimizers import RMSprop
```

Split and resize:

```
import os

import random

import shutil
```

```

from PIL import Image

import cv2

root_dir = "RawDataset"


train_path = "Dataset/train"

val_path = "Dataset/val"


# Set the percentage of images to use for each set

train_ratio = 0.8


val_ratio = 0.3


# loop through each class folder in the original dataset
for class_folder in os.listdir(root_dir):

    class_folder_path = os.path.join(root_dir, class_folder)


    # create corresponding class folders in the destination folders

    train_class_path = os.path.join(train_path, class_folder)

    os.makedirs(train_class_path, exist_ok=True)

    val_class_path = os.path.join(val_path, class_folder)

    os.makedirs(val_class_path, exist_ok=True)


# shuffle the images and split them into training, testing, and validation sets

images = os.listdir(class_folder_path)

random.shuffle(images)

```

```

train_images = images[:int(train_ratio*len(images))]

val_images = images[int(val_ratio*len(images)):]

# resize and save the images to their respective destination folders

for image in train_images:

    image_path = os.path.join(class_folder_path, image)

    img = Image.open(image_path)

    img = img.resize((224, 224)) # replace 224 with desired size

    img.save(os.path.join(train_class_path, image))

for image in val_images:

    image_path = os.path.join(class_folder_path, image)

    img = Image.open(image_path)

    img = img.resize((224, 224)) # replace 224 with desired size

    img.save(os.path.join(val_class_path, image))

```

Assign Image size and Training Rate:

```

warnings.filterwarnings('ignore')

IMAGE_SIZE = (128,128)

INPUT_SHAPE = (128,128, 3)

TRAINING_RATE = 0.0001

```

Load Training and Testing Datasets to generate Model:

```

train_datagen = ImageDataGenerator()

validate_datagen = ImageDataGenerator()

```

```

training_set = train_datagen.flow_from_directory(
    'Dataset/train',
    target_size = IMAGE_SIZE,
    class_mode = 'categorical',
    batch_size = 20)

validation_set = validate_datagen.flow_from_directory(
    'Dataset/val',
    target_size = IMAGE_SIZE,
    class_mode = 'categorical',
    batch_size = 20)

classes = {v: k for k, v in training_set.class_indices.items()}

classes

```

Initialize Training method and Training Rate:

```

#Model

input_shape = INPUT_SHAPE

inputs = Input(input_shape)

input_tensor = Input(shape= (128,128,3))

#xception = Xception(include_top=False, input_shape = input_shape,)(inputs)

xception = Xception(include_top=False, input_shape = input_shape, weights =
'imagenet')(inputs)

res_net = ResNet50(input_tensor = input_tensor, include_top = False, weights =
'imagenet')(inputs)

```

```

#output

outputs      =      Concatenate(axis=-1)([GlobalAveragePooling2D()(xception),
GlobalAveragePooling2D()(res_net)])

#outputs = Concatenate(axis=-1)([GlobalAveragePooling2D()(xception)])

outputs = Dropout(0.25)(outputs)

outputs = Dense(len(classes), activation='softmax')(outputs)


# Set Optimizer

opt1 = SGD(lr=TRAINING_RATE)
opt2 = RMSprop(lr=TRAINING_RATE)


model = Model(inputs, outputs)

model.compile(optimizer=opt1,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

```

Visualize the Trained model accuracy:

```

fig, ax = plt.subplots(1,2, figsize=(10,5))

ax = ax.ravel()

for i, met in enumerate(['accuracy','loss']):

    ax[i].plot(Training_model.history[met])

    ax[i].set_title('Model {}'.format(met))

    ax[i].set_xlabel('epochs')

```

```

ax[i].set_ylabel(met)

ax[i].legend(['train'])

plt.show()

plt.plot(Training_model.history['accuracy'])
plt.plot(Training_model.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarize history for loss

plt.plot(Training_model.history['loss'])
plt.plot(Training_model.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

```

Confusion Matrix:

```

import tensorflow as tf

import numpy as np

```

```

from sklearn.metrics import confusion_matrix

import os

# Load the saved model

model = tf.keras.models.load_model('Trained_models/LastModel.h5')

# Get the testing dataset directory

test_dir = 'rawdataset'

# Get the labels from the testing folder names

labels = os.listdir(test_dir)

print('Labels:', labels)

# Create the testing dataset using ImageDataGenerator

test_datagen = tf.keras.preprocessing.image.ImageDataGenerator()

test_generator = test_datagen.flow_from_directory(

    test_dir,

    target_size=(128,128),

    batch_size=15,

    class_mode='categorical',

    shuffle=False)

# Get the predicted classes for the testing dataset

predictions = model.predict(test_generator)

predicted_classes = np.argmax(predictions, axis=1)

# Get the true classes for the testing dataset

true_classes = test_generator.classes

# Create the confusion matrix

cm = confusion_matrix(true_classes, predicted_classes)

```

```

# Print the confusion matrix

print('Confusion matrix:\n', cm)

# Print the accuracy

accuracy = np.sum(np.diagonal(cm)) / np.sum(cm)

print('Accuracy:', accuracy)

# Print the labels

print('Labels:', labels)

tp = np.diag(cm)

tn = np.sum(cm) - np.sum(tp)

print(f'True Positives: {tp}')

print(f'True Negatives: {tn}')

#Specificity = True Negative / (True Negative + False Positive)

#Precision = True Positive / (True Positive + False Positive)

#Recall = True Positive / (True Positive + False Negative)

# Calculate specificity, precision and recall

specificity = cm[0,0]/(cm[0,0]+cm[0,1])

precision = cm[1,1]/(cm[1,1]+cm[0,1])

recall = cm[1,1]/(cm[1,1]+cm[1,0])

print("Specificity: ", specificity)

print("Precision: ", precision)

print("Recall: ", recall)

```

Model Deployment:

Model Deployment

```
import numpy as np

import streamlit as st

from PIL import Image

import keras.preprocessing.image as image

from tensorflow.keras import models

from PIL import Image


# Define utility functions

def load_saved_model(model_path):

    saved_model = models.load_model(model_path, custom_objects={"accuracy":
"accuracy"})

    return saved_model


def preprocess_uploaded_image(image_file, desired_size=128):

    image_data = image_file.resize((desired_size, desired_size),
resample=Image.LANCZOS)

    image_data = np.expand_dims(image_data, axis=0) # Add batch dimension

    return image_data


def resize_image(image_file, desired_size=254):

    resized_image = image_file.resize((desired_size, desired_size))

    return resized_image


def generate_custom_footer():
```

```

custom_footer_html = ""

<div style="background-color:#F8F9FA;padding:10px;border-radius:10px;">

<p style="text-align:center;">Developed by: Srinivas Tejaswi</p>

</div>

""

return custom_footer_html

```

Main Streamlit code

```
highlight_prediction_colors = ["red", "orange", "green", "brown", "blue", "purple"]
```

```
disease_types = [
```

```
    'Apple Black rot',
```

```
    'Apple Cedar apple rust',
```

```
    'Corn (maize) Common rust',
```

```
    'Corn (maize) Northern Leaf Blight',
```

```
    'Tomato Tomato Yellow Leaf Curl Virus',
```

```
    'Tomato Tomato mosaic virus'
```

```
]
```

```
disease_suggestions = {
```

```
    'Apple Black rot': ['Apply fungicide', 'Prune affected areas', 'Improve air circulation'],
```

```
    'Apple Cedar apple rust': ['Prune affected areas', 'Remove fallen leaves', 'Apply fungicide'],
```

```
    'Corn (maize) Common rust': ['Rotate crops', 'Plant resistant varieties', 'Use fungicides'],
```

```
'Corn (maize) Northern Leaf Blight': ['Plant resistant varieties', 'Rotate crops', 'Use fungicides'],
```

```
'Tomato Tomato Yellow Leaf Curl Virus': ['Remove infected plants', 'Use insecticides', 'Plant resistant varieties'],
```

```
'Tomato Tomato mosaic virus': ['Remove infected plants', 'Control insect vectors', 'Use virus-free seeds']
```

```
}
```

```
def format_highlighted_prediction(disease_types, idx):
```

```
    formatted_disease_types = disease_types.copy()
```

```
    highlighted_prediction_html = f"<span style='color:{highlight_prediction_colors[idx]}; font-size: larger">{disease_types[idx]}</span>"
```

```
    formatted_disease_types[idx] = highlighted_prediction_html
```

```
    return '<br>'.join(formatted_disease_types)
```

```
if __name__ == '__main__':
```

```
    st.set_page_config(page_title="Integrated Plant Disease Detection System",  
page_icon="🌿")
```

```
    st.markdown("<style>footer{visibility:hidden;}</style>", unsafe_allow_html=True)
```

```
    st.markdown("<br>", unsafe_allow_html=True)
```

```
    st.markdown("<h1 style='text-align: center; color: black;'>"
```

```
        "<center>&emsp;&emsp;Integrated Plant Disease Detection  
System</center></h1>", unsafe_allow_html=True)
```

```
    st.markdown("<br>", unsafe_allow_html=True)
```

```
    uploaded_file = st.file_uploader("Upload a Leaf image...", type=["jpg", "png",  
"jpeg"])
```

```

if uploaded_file is not None:

    img_in = Image.open(uploaded_file)

    resized_img_in = resize_image(img_in) # Resize uploaded image

    img_in_processed = preprocess_uploaded_image(resized_img_in) #
Preprocess for prediction

col1, col2 = st.columns(2)

col1.image(resized_img_in)

st.write("Leaf Image")


model = load_saved_model('Trained_Models/LastModel.h5')

prediction = model.predict(img_in_processed)

predicted_idx = np.argmax(prediction)

col2.markdown("### Predicted Type")


col2.markdown(format_highlighted_prediction(disease_types, predicted_idx),
unsafe_allow_html=True)


if disease_types[predicted_idx] in disease_suggestions:

    st.write("### Suggestions:")

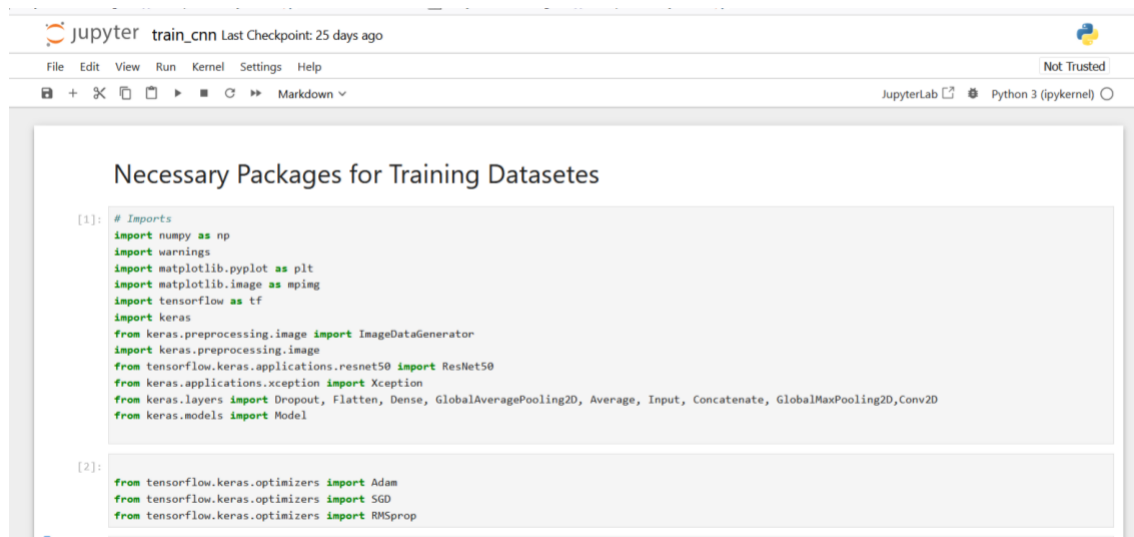
    for suggestion in disease_suggestions[disease_types[predicted_idx]]:

        st.write(f"- {suggestion}")


st.markdown(generate_custom_footer(), unsafe_allow_html=True)

```

B. SCREENSHOTS

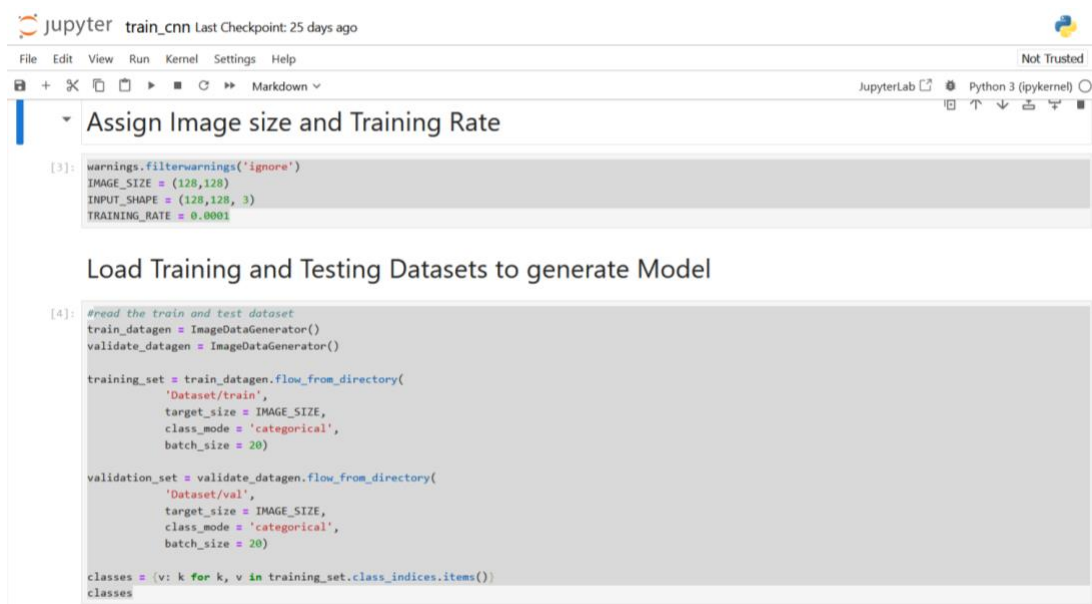


The screenshot shows a JupyterLab window titled 'train_cnn' with a 'Last Checkpoint: 25 days ago' status. The interface includes a top menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and execution. The main area displays a code cell with the title 'Necessary Packages for Training Datasets'. The code in the cell imports various libraries: numpy, warnings, matplotlib.pyplot, matplotlib.image, tensorflow, keras, and specific models and layers from tensorflow.keras and keras. The code is as follows:

```
[1]: # Imports
import numpy as np
import warnings
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import tensorflow as tf
import keras
from keras.preprocessing.image import ImageDataGenerator
import keras.preprocessing.image
from tensorflow.keras.applications.resnet50 import ResNet50
from keras.applications.xception import Xception
from keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D, Average, Input, Concatenate, GlobalMaxPooling2D, Conv2D
from keras.models import Model

[2]:
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers import RMSprop
```

Fig B.1 Importing necessary packages



The screenshot shows a JupyterLab window titled 'train_cnn' with a 'Last Checkpoint: 25 days ago' status. The interface includes a top menu bar (File, Edit, View, Run, Kernel, Settings, Help) and a toolbar with icons for file operations and execution. The main area displays a code cell with the title 'Assign Image size and Training Rate'. The code in the cell sets the image size and training rate. The code is as follows:

```
[3]: warnings.filterwarnings('ignore')
IMAGE_SIZE = (128,128)
INPUT_SHAPE = (128,128, 3)
TRAINING_RATE = 0.0001
```

Below this code cell, there is another code cell with the title 'Load Training and Testing Datasets to generate Model'. The code in this cell loads the training and testing datasets using ImageDataGenerator and flow_from_directory. The code is as follows:

```
[4]: #read the train and test dataset
train_datagen = ImageDataGenerator()
validate_datagen = ImageDataGenerator()

training_set = train_datagen.flow_from_directory(
    'Dataset/train',
    target_size = IMAGE_SIZE,
    class_mode = 'categorical',
    batch_size = 20)

validation_set = validate_datagen.flow_from_directory(
    'Dataset/val',
    target_size = IMAGE_SIZE,
    class_mode = 'categorical',
    batch_size = 20)

classes = [v: k for k, v in training_set.class_indices.items()]
classes
```

Fig B.2 Assigning Size, Training and Testing Datasets

jupyter train_cnn Last Checkpoint: 25 days ago

File Edit View Run Kernel Settings Help

Not Trusted

JupyterLab Python 3 (ipykernel)

Initialize Training method and Training Rate

```
[6]: #Model
input_shape = INPUT_SHAPE
inputs = Input(input_shape)
input_tensor = Input(shape=(128,128,3))

#xception = Xception(include_top=False, input_shape = input_shape)(inputs)
xception = Xception(include_top=False, input_shape = input_shape, weights = 'imagenet')(inputs)
res_net = ResNet50(input_tensor, include_top = False, weights = 'imagenet')(inputs)

#output
outputs = Concatenate(axis=-1)(GlobalAveragePooling2D()(xception), GlobalAveragePooling2D()(res_net))
#outputs = Concatenate(axis=-1)(GlobalAveragePooling2D()(xception))
outputs = Dropout(0.25)(outputs)
outputs = Dense(len(classes), activation='softmax')(outputs)

# Set Optimizer
opt1 = SGD(lr=TRAINING_RATE)
opt2 = RMSProp(lr=TRAINING_RATE)

model = Model(inputs, outputs)
model.compile(optimizer=opt1,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Fig B.3 Initialize Training methods and Training Rate

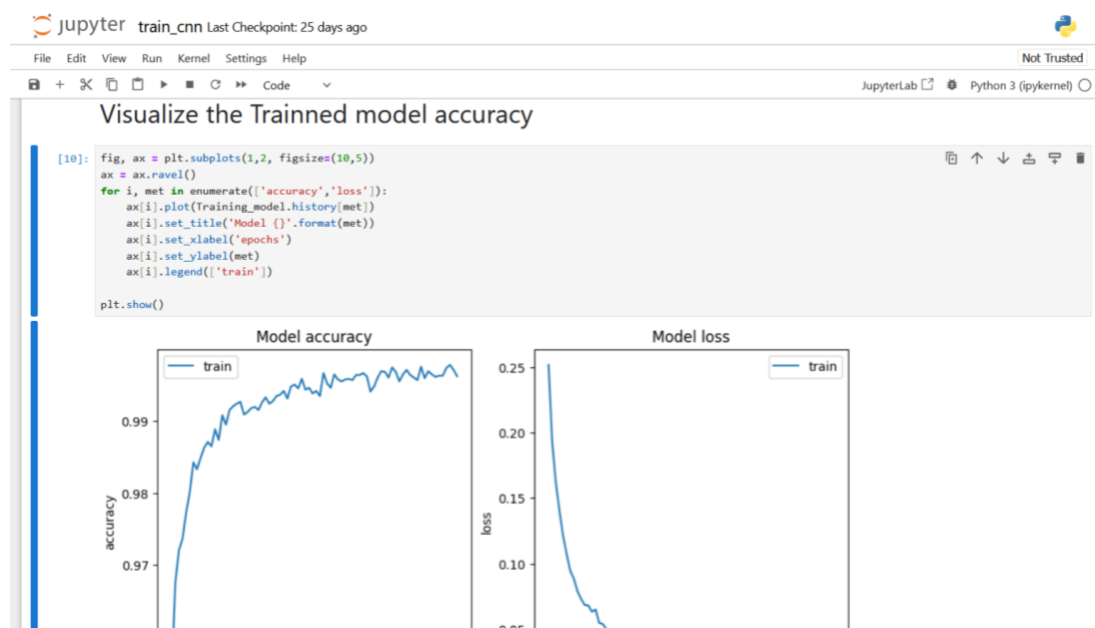


Fig B.4 Model Accuracy

```

Jupyter train_cnn Last Checkpoint: 25 days ago
File Edit View Run Kernel Settings Help
JupyterLab Python 3 (ipykernel)

[11]: plt.plot(Training_model.history['accuracy'])
      plt.plot(Training_model.history['val_accuracy'])
      plt.title('model accuracy')
      plt.ylabel('accuracy')
      plt.xlabel('epoch')
      plt.legend(['train', 'test'], loc='upper left')
      plt.show()

      # summarize history for loss

      plt.plot(Training_model.history['loss'])
      plt.plot(Training_model.history['val_loss'])
      plt.title('model loss')
      plt.ylabel('loss')
      plt.xlabel('epoch')
      plt.legend(['train', 'test'], loc='upper left')
      plt.show()

```

Fig B.5 Accuracy

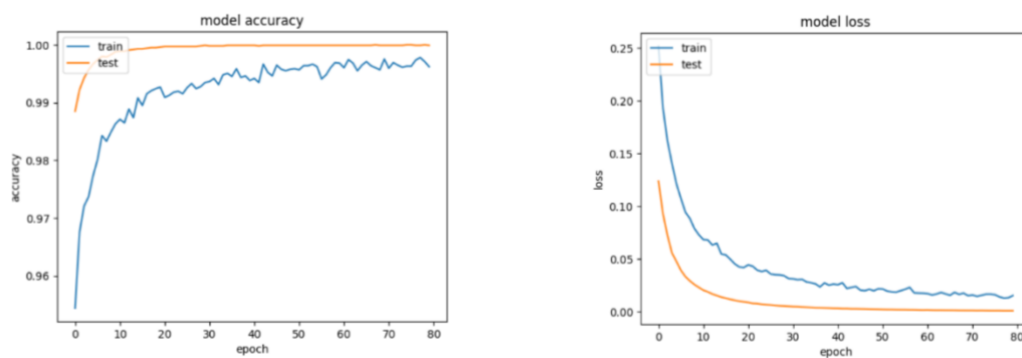


Fig B.6 Accuracy and Loss of the Model

```

Jupyter Split and Resize Last Checkpoint: 2 months ago
File Edit View Run Kernel Settings Help
JupyterLab Python 3 (ipykernel)

[1]: import os
      import random
      import shutil
      from PIL import Image
      import cv2

[2]: # Set the root directory of your dataset
      root_dir = "RawDataset"

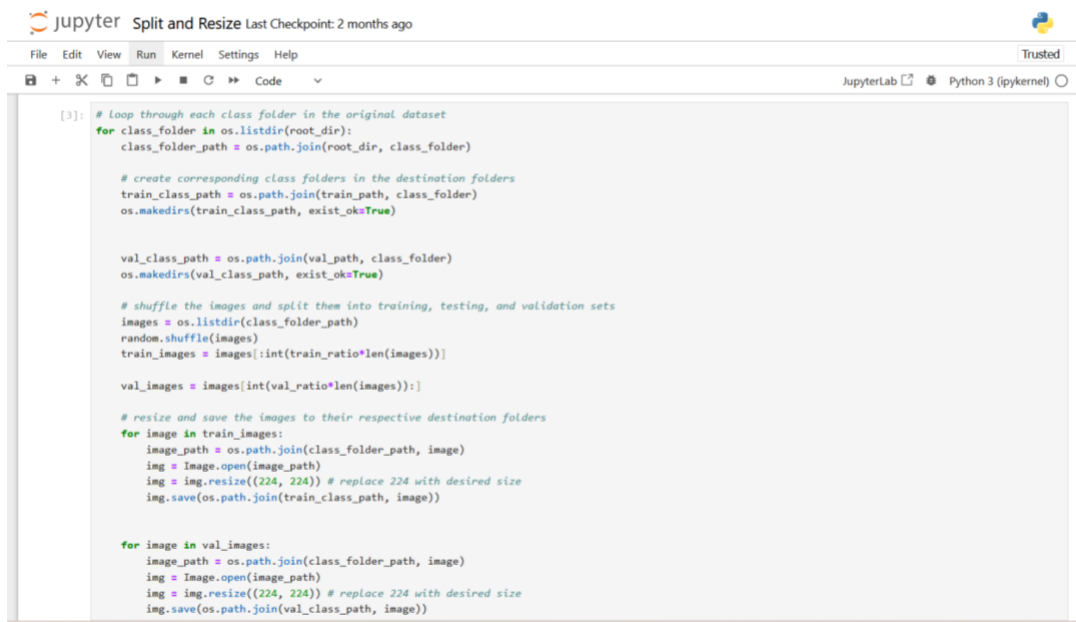
      train_path = "Dataset/train"
      val_path = "Dataset/val"

      # Set the percentage of images to use for each set
      train_ratio = 0.8

      val_ratio = 0.3

```

Fig B.7



The image shows a JupyterLab interface with a code cell containing Python code for splitting and resizing images. The code is as follows:

```
[3]: # Loop through each class folder in the original dataset
for class_folder in os.listdir(root_dir):
    class_folder_path = os.path.join(root_dir, class_folder)

    # create corresponding class folders in the destination folders
    train_class_path = os.path.join(train_path, class_folder)
    os.makedirs(train_class_path, exist_ok=True)

    val_class_path = os.path.join(val_path, class_folder)
    os.makedirs(val_class_path, exist_ok=True)

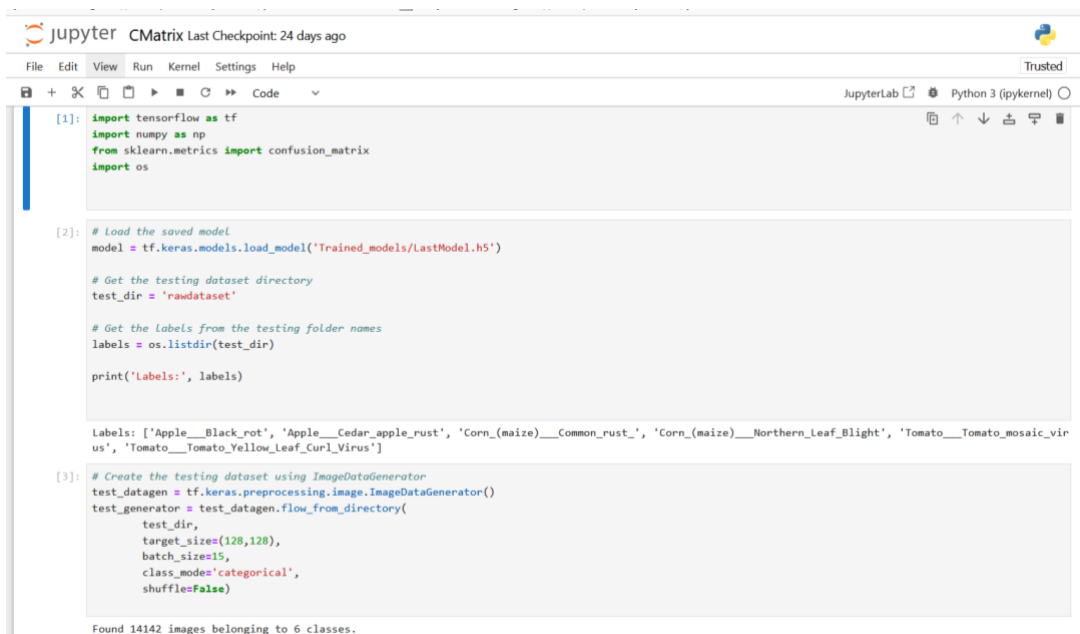
    # shuffle the images and split them into training, testing, and validation sets
    images = os.listdir(class_folder_path)
    random.shuffle(images)
    train_images = images[:int(train_ratio*len(images))]

    val_images = images[int(val_ratio*len(images)):]

    # resize and save the images to their respective destination folders
    for image in train_images:
        image_path = os.path.join(class_folder_path, image)
        img = Image.open(image_path)
        img = img.resize((224, 224)) # replace 224 with desired size
        img.save(os.path.join(train_class_path, image))

    for image in val_images:
        image_path = os.path.join(class_folder_path, image)
        img = Image.open(image_path)
        img = img.resize((224, 224)) # replace 224 with desired size
        img.save(os.path.join(val_class_path, image))
```

Fig B.8 Splitting and Resize



The image shows a JupyterLab interface with a code cell containing Python code for loading a model and creating a testing dataset. The code is as follows:

```
[1]: import tensorflow as tf
import numpy as np
from sklearn.metrics import confusion_matrix
import os

[2]: # Load the saved model
model = tf.keras.models.load_model('Trained_models/LastModel.h5')

# Get the testing dataset directory
test_dir = 'rawdataset'

# Get the Labels from the testing folder names
labels = os.listdir(test_dir)

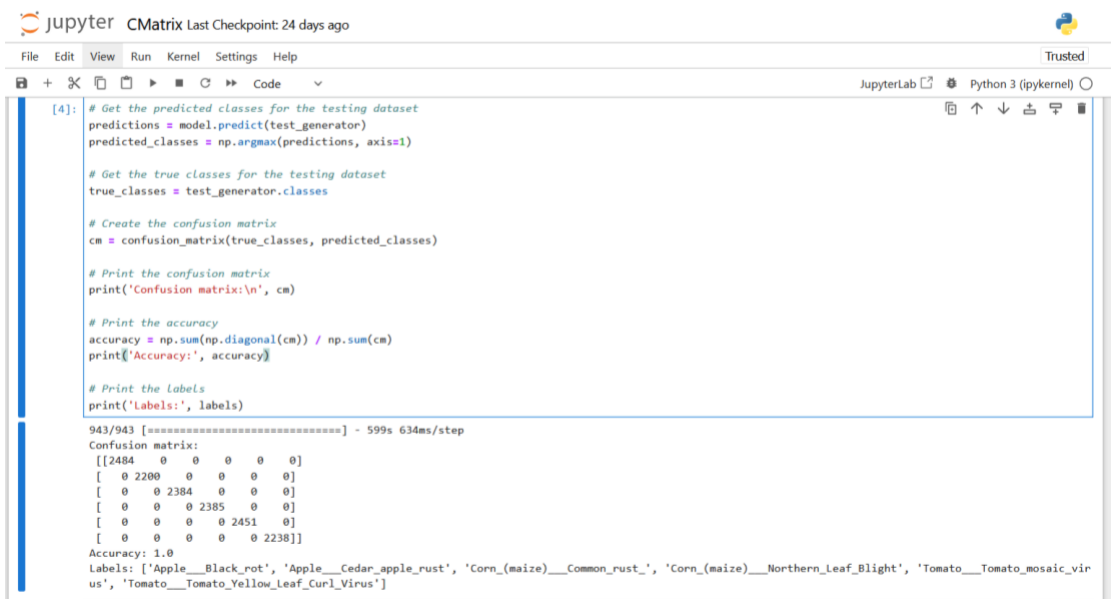
print('Labels:', labels)

Labels: ['Apple__Black_rot', 'Apple__Cedar_apple_rust', 'Corn_(maize)__Common_rust_', 'Corn_(maize)__Northern_Leaf_Blight', 'Tomato__Tomato_mosaic_virus', 'Tomato__Tomato_Yellow_Leaf_Curl_Virus']

[3]: # Create the testing dataset using ImageDataGenerator
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator()
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(128,128),
    batch_size=15,
    class_mode='categorical',
    shuffle=False)

Found 14142 images belonging to 6 classes.
```

Fig B.9 CMatrix



```

[4]: # Get the predicted classes for the testing dataset
predictions = model.predict(test_generator)
predicted_classes = np.argmax(predictions, axis=1)

# Get the true classes for the testing dataset
true_classes = test_generator.classes

# Create the confusion matrix
cm = confusion_matrix(true_classes, predicted_classes)

# Print the confusion matrix
print('Confusion matrix:\n', cm)

# Print the accuracy
accuracy = np.sum(np.diagonal(cm)) / np.sum(cm)
print('Accuracy:', accuracy)

# Print the Labels
print('Labels:', labels)

943/943 [=====] - 599s 634ms/step
Confusion matrix:
[[2484  0  0  0  0  0]
 [ 0 2200  0  0  0  0]
 [ 0  0 2384  0  0  0]
 [ 0  0  0 2385  0  0]
 [ 0  0  0  0 2451  0]
 [ 0  0  0  0  0 2238]]
Accuracy: 1.0
Labels: ['Apple__Black_rot', 'Apple__Cedar_apple_rust', 'Corn_(maize)__Common_rust_', 'Corn_(maize)__Northern_Leaf_Blight', 'Tomato__Tomato_mosaic_virus', 'Tomato__Tomato_Yellow_Leaf_Curl_Virus']

```

Fig B.10 Confusion Matrix Analysis



```

[5]: tp = np.diag(cm)
tn = np.sum(cm) - np.sum(tp)
print(f'True Positives: {tp}')
print(f'True Negatives: {tn}')

#Specificity = True Negative / (True Negative + False Positive)
#Precision = True Positive / (True Positive + False Positive)
#Recall = True Positive / (True Positive + False Negative)

# Calculate specificity, precision and recall
specificity = cm[0,0]/(cm[0,0]+cm[0,1])
precision = cm[1,1]/(cm[1,1]+cm[0,1])
recall = cm[1,1]/(cm[1,1]+cm[1,0])

print("Specificity: ", specificity)
print("Precision: ", precision)
print("Recall: ", recall)

True Positives: [2484 2200 2384 2385 2451 2238]
True Negatives: 0
Specificity: 1.0
Precision: 1.0
Recall: 1.0

```

Fig B.11 Precision Values

B.1 Implementation Screenshots:



Fig B.12 Front Page

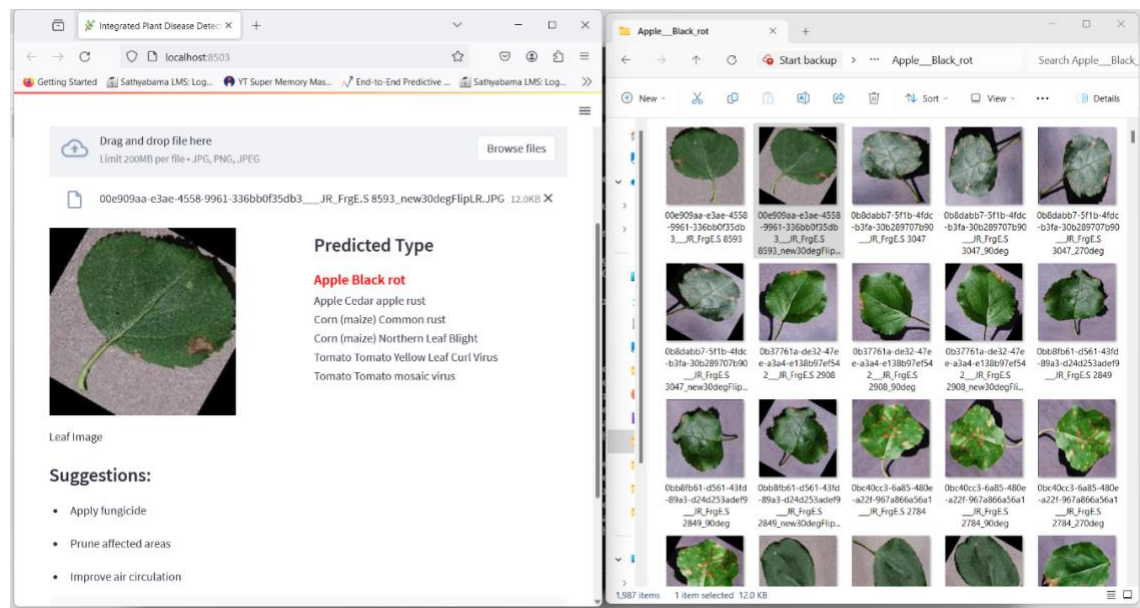


Fig B.13 Uploading and fetching output

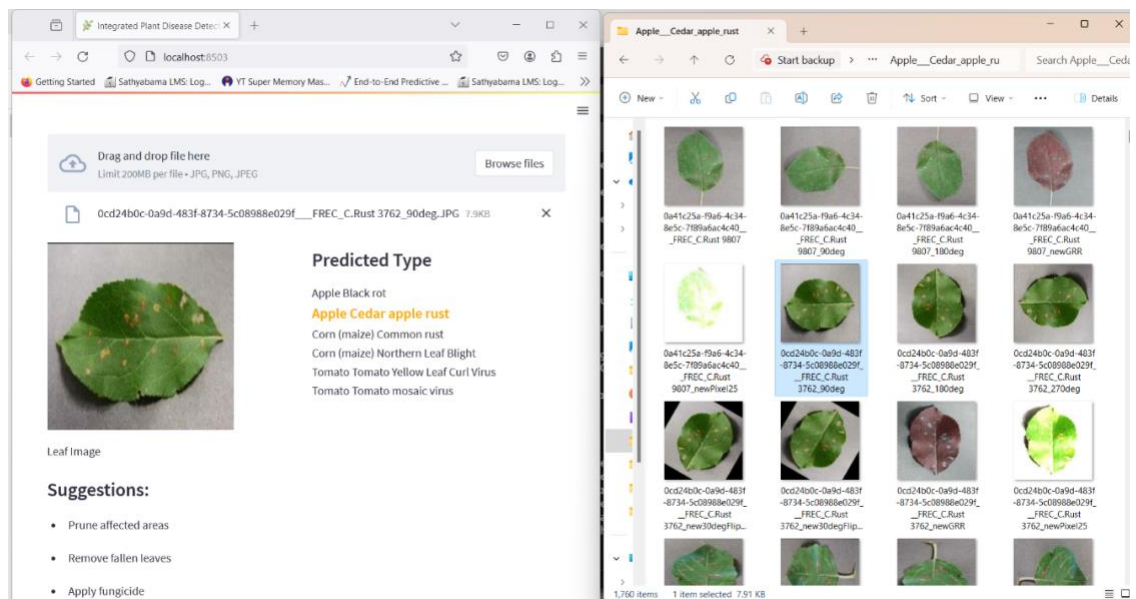


Fig B.14

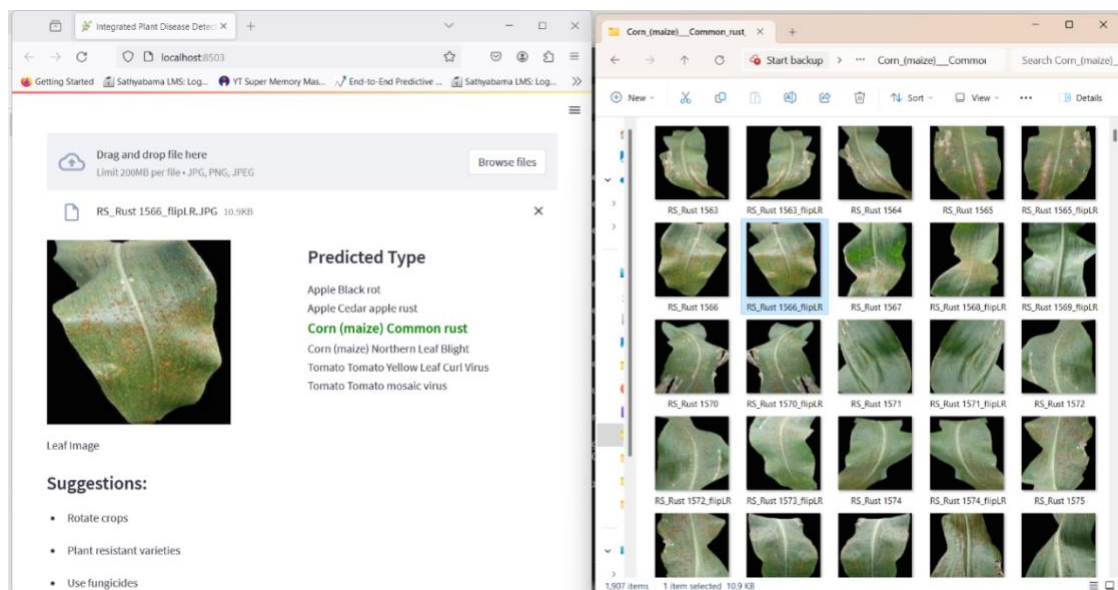


Fig B.15

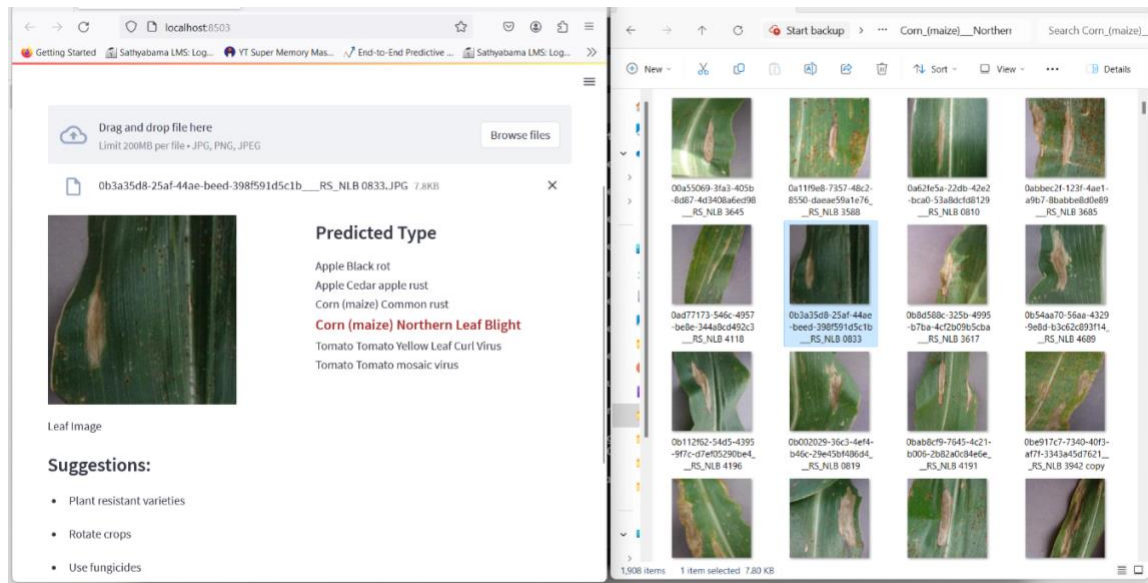


Fig B.16

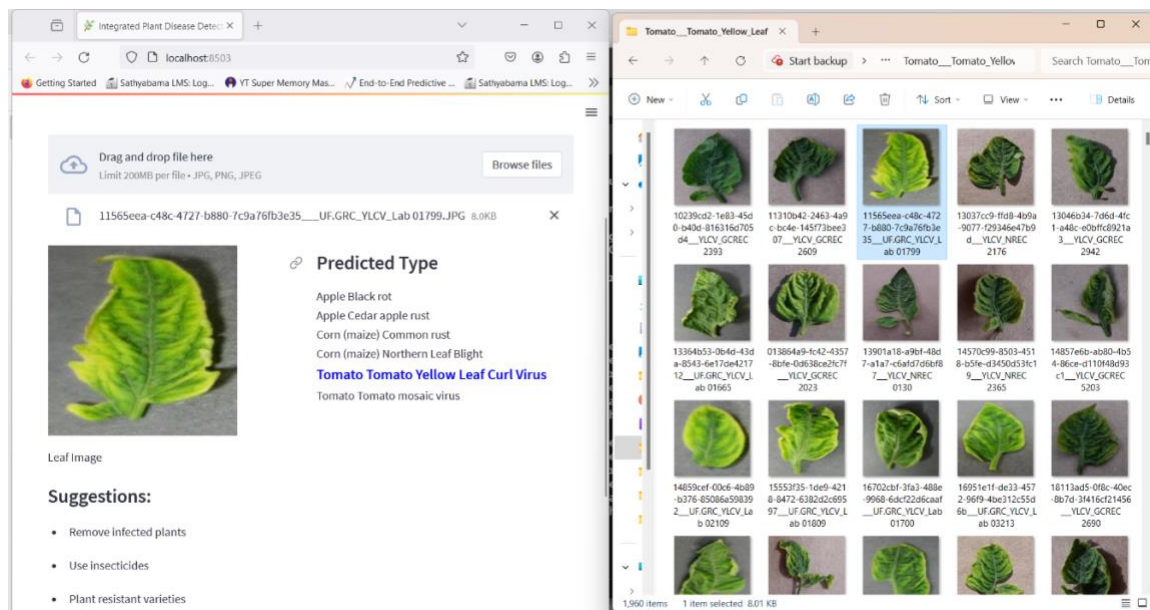


Fig B.17

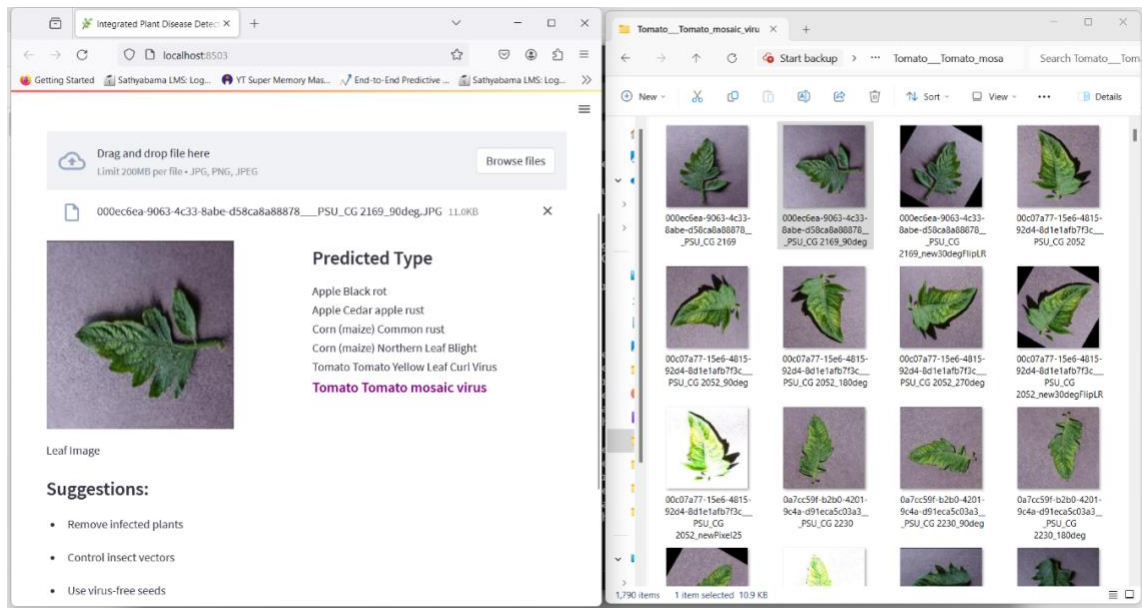


Fig B.18

C. RESEARCH PAPER

INTEGRATED PLANT DISEASE DETECTION SYSTEM

R. Aishwarya

R. Srinivas

R. Tejaswi

Assistant Professor

Student

Student

Department of Computer Science and
Engineering

Department of Computer Science and
Engineering

Department of Computer Science and
Engineering

Sathyabama Institute of Science &
Technology, Chennai, India

Sathyabama Institute of Science &
Technology, Chennai, India

Sathyabama Institute of Science &
Technology, Chennai, India

[aishwaryarajkumar7@sathyabama.ac.i
n](mailto:aishwaryarajkumar7@sathyabama.ac.in)

srinivas622003@gmail.com

tejaswiramesh22@gmail.com

ABSTRACT---Nowadays crops are affected by different types of diseases which farmers cannot find what disease it is. It takes more time to know about the affected disease and finding cure for the disease is a long time process. Within that time the crop will be affected by the disease severely or it may die. To make this disease detection process more easier we have introduced our web application or mobile application which makes the disease detection process more easier and it takes less amount of time. This can help farmers as well as people to easily detect which type of disease is affected to the crop and suggest the best ways to overcome the issues. This process is cost efficient when compared to other techniques. This system introduces an Automated Plant Disease Identification System using hybrid Deep Learning model, which involves merging of Xception and ResNet50 architectures. Focused on apples, corn, and tomatoes, the system aids farmers in prompt and precise disease diagnosis. This also helps people to do agriculture in more efficient and easy ways in small area. Leveraging diverse datasets, the model classifies diseases such as apple, corn and tomato. Stochastic gradient descent optimizes model performance, facilitating practical deployment through a user friendly web application. The system aims to enhance agriculture efficiency, contribute to sustainable practice, and ensure food security by addressing the challenges posed by plant diseases.

KEYWORDS:

Hybrid, Xception, ResNet50, Agriculture Management, Web Applications, Food Security.

I. INTRODUCTION

Diseases in the plants present a huge threat to global agriculture and also impacting crop yield, food security, and stability of the economy. The precise and accurate identification of these diseases at the perfect time is crucial for effective crop management and disease mitigation. Traditional disease diagnosis method in agriculture often rely on inspection by farmers visually, which can not be very

accurate and prone to errors. In response to these challenges, this work proposes an Automated Plant Disease Identification System that harnesses the power of machine learning, specifically a hybrid deep learning model amalgamating Xception and ResNet50 architectures.

The escalating global demand for food necessitates innovative solutions to protect crop from diseases and enhance agriculture productivity. By integrating advanced technologies into the existing agriculture framework, we aim to provide farmers and agricultural professionals with a reliable and efficient tool for the rapid diagnosis of plant diseases.

In this introduction, we will discuss the significance of addressing plant diseases, the limitations of current manual methods, and the potential impact of introducing an automated system. Additionally, we will provide an overview of the objectives, scope, and structure of the proposed system, emphasizing the role of technology in revolutionizing plant diseases identification and contributing to sustainable agriculture.

II. LITERATURE REVIEW

Muhammad Hammad Saleem et.al.[1] implemented a deep learning model to identify disease affected the crops by comparing all the available techniques and found the best technique for their work. They used the best technique called Xception which accrued high accuracy and it was trained to identify 26 diseases in 14 different plants. They used this deep learning technique to identify plant disease which then resulted in accuracy of 99.8%.

Janani Varthini et.al.[2] suggested a deep learning model to identify the disease in the cotton plant combining two

effective CNN architectures called Xception and ResNet50 by using a large amount of dataset so that farmers will find it easier to identify disease in cotton plant. They acquired an accuracy of 98%.

Vandana B. Malode et.al.[3] introduced a deep learning model to identify the disease infected the crops especially tomato and grapes using the (CNN)Convolutional Neural Network method which is used for detecting Multiple Crops Leaf Disease (MCLD).The CNN based architectural network known as Visual Geometry Group(VSG) was used by them. They used these techniques to show precise and accurate results for the crops and accrued an accuracy of 98.40%.

G.Sumalatha et.al.[4] presented a deep learning model to identify diseases in plants using the Convolutional Neural Network(CNN) and Deep Neural Network(DNN).Their goal was to create a system which can identify the disease using a simple leaf image. They used six different CNN architectures to implement the system from which DenseNet21 Architecture acquired an higher accuracy of 95.48% by detecting eight types of diseases.

Usman Abdur Rehman et.al.[5] using Deep Learning techniques he proposed a system called ResNet50 to identify diseases in corn, potatoes and Tomatoes. The ResNet50 performance was very high when compared with other two models, VGG16 and MCNN, which was built and also was trained from the beginning. ResNet50 gave the best results, achieving 98.7% accuracy in predicting the plant disease.

Sakshi Raina et. al. [6] Presented a deep learning Technique model to detect plant leaf disease using the images of the leaf. They used artificial intelligence and machine learning to analyse plant leaf images. They performed various key steps like image acquisition, image preprocessing etc.. based on classifier to identify the disease. They proposed a system to detect plant disease quickly and accurately from pictures.

III. METHODOLOGY

Automated Plant Disease Identification System

A .Data Collection and Preparation

Careful compilation of a diverse dataset featuring labelled images showcasing various stages of plant diseases affecting apples, corn, and tomatoes.

Application of meticulous preprocessing techniques to standardize images, ensuring uniformity in resolution, colour, and quality.

B. Model Architecture Selection and Fusion

Selection of appropriate deep learning architectures post data preparation, considering the unique characteristics of diseases in each crop.

Fusion of selected architectures into a hybrid model to leverage the individual strengths for recognizing complex disease patterns.

Implementation using the most familiar DL frameworks like TensorFlow and PyTorch for efficient computation and flexibility for the training of the model.

C. Training and Optimization

Training of the hybrid DL model using the available dataset is performed. Learning rate, batch size, and optimizer setting parameters are fine tuned to acquire higher model performance.

Usage of techniques such as the stochastic gradient descent and methods of regularization are performed to prevent the overfitting and enhance the generalization of the model.

Iterative training-validation are performed to monitor the convergence and to upgrade the performance metrics.

D. Performance Evaluation and Validation

The Validation phase is used to assess the performance of the model for its unseen data segments.

Computation of evaluation metrics such as accuracy, precision, recall, and F1-score are performed to quantify the proficiency of the model.

Usage of the Cross-validation techniques and also the confusion matrix analysis to calculate the robustness of the model and also to identify the weakness of the model.

E. Integration and Deployment

Integration of the model into a web application that is user friendly using Streamlit upon achieving satisfactory performance.

Development of an intuitive interface enabling farmers and agricultural professionals to seamlessly upload images of plant samples for rapid and accurate disease predictions.

Strong testing is done to ensure the stability and compatibility of the application across variety of platforms.

Incorporation of feedback from users to refine the usability of the application, ensuring seamless integration into agricultural practices.

IV. METHODS AND TECHNIQUES USED

Automated Plant Disease Identification System

A .Deep Learning Architectures:

Utilization of deep learning architectures, such as Xception and ResNet50, for effective feature extraction and pattern recognition in plant images.

B. Transfer Learning

Implementation of the familiar transfer learning techniques to leverage the already trained models (Xception and ResNet50) for the efficient training of the hybrid model.

C. Optimization Techniques:

Fine tuning of the hybrid DL model using the available dataset is performed. Learning rate,batch size, and optimizer

setting parameters are fine tuned to acquire higher model performance.

D. Stochastic Gradient Descent:

Use of stochastic gradient descent as an optimization algorithm to iteratively update model weights, preventing overfitting and enhancing generalization capabilities.

E. Regularization Methods:

Usage of the regularisation techniques such as dropout or L2 regularization, to control the overfitting of the data and improve the ability of the model to generalize the unseen data.

F. Cross-Validation:

Cross-validation techniques that includes k-fold cross-validation, to evaluate model robustness and identify potential weaknesses.

G. Confusion Matrix Analysis:

Utilization of confusion matrix analysis to provide insights into model performance, identifying areas of strengths and weaknesses in disease classification.

H. Streamlit for Web Application Development:

Adoption of Streamlit, a Python library, for the development of an intuitive and user-friendly web application for seamless model integration and user interaction.

V. EXISTING SYSTEM:

The current manual inspection for plant disease diagnosis is subjective and labour-intensive. It introduces variability in results and may lead to delays in disease identification, impacting agricultural productivity. Subjectivity in disease diagnosis due to reliance on manual inspection methods. Human errors, influenced by factors like fatigue, impact the accuracy and consistency of disease diagnoses. Resource-intensive nature and limited scalability, particularly challenging with large datasets or a high volume of plant samples. Efficiency hindered by the heavy dependence on the expertise of individual farmers or agricultural professionals. Lack of standardized protocols and criteria for disease diagnosis leading to variations in reporting and diagnostic criteria.

VI. PROPOSED SYSTEM:

The proposed automated system employs a hybrid deep learning model for efficient and accurate plant disease identification. It aims to enhance the speed, precision, and scalability of disease diagnosis, reducing dependency on manual inspection and fostering sustainable agricultural practices.

The primary aim is to implement an advanced and automatic system for the identification of plant diseases affecting apples, corn, and tomatoes. Leveraging state-of-the-art deep learning techniques, the system targets the precise and rapid classification of various plant diseases, providing farmers and

agricultural professionals with an efficient tool for early detection and intervention. The overarching goal is to enhance agricultural productivity and sustainability by significantly reducing the reliance on manual inspection methods and facilitating timely and accurate disease diagnosis. Through the integration of machine learning models and user-friendly interfaces, the system aims to empower users with accessible and actionable insights, affording to improvised crop management and, consequently, the overall well-being of the agricultural ecosystem.

Automation for efficient and consistent plant disease identification. Improvisation of accuracy with the familiar and advanced ML models. Rapid processing for accurate interventions in agriculture. Enhanced scalability for handling large datasets. Reducing the dependency on individual expertise, making it accessible to a broader user base. User-friendly interface to increase the accessibility among the farmers. Often improvement through collaboration with agricultural experts. Drastic Enhancement of agriculture productivity and sustainability.

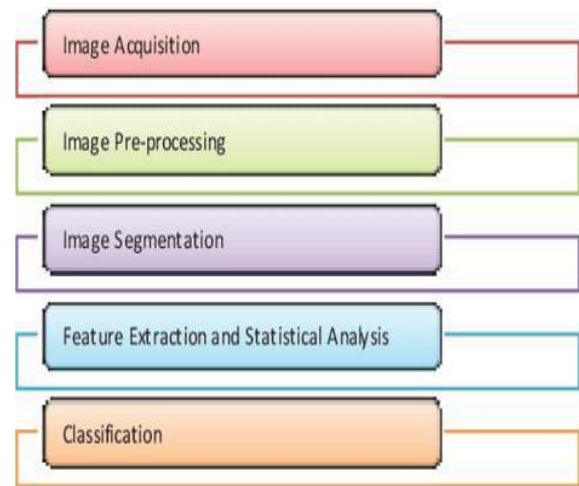


Fig 1.1

A. Image Acquisition: Image acquisition is done using the mobile phones, cameras to capture the images of the leaves.

B. Image Pre-processing: After stage i) that is image acquisition, if the image is not as it is expected or the area of interest are not clear we use this image preprocessing. Cleansing, integration, transformation are some of its methods.

C. Image Segmentation: In this process the image is split into separate parts based on same characteristics. As a result, we get only the parts which have same characteristics as of in the full image.

D. Feature Extraction and Statistical Analysis: Feature Extraction means extracting relevant information from raw

data to implement it more effectively. Statistical Analysis is used to analyze mathematical trend to analyze , interpret or relationships between data. Together they are important to extract meaningful insights from complex dataset.

E. Classification: This technique is used in training data to identify the types of new observations.

VII. FLOWCHART:

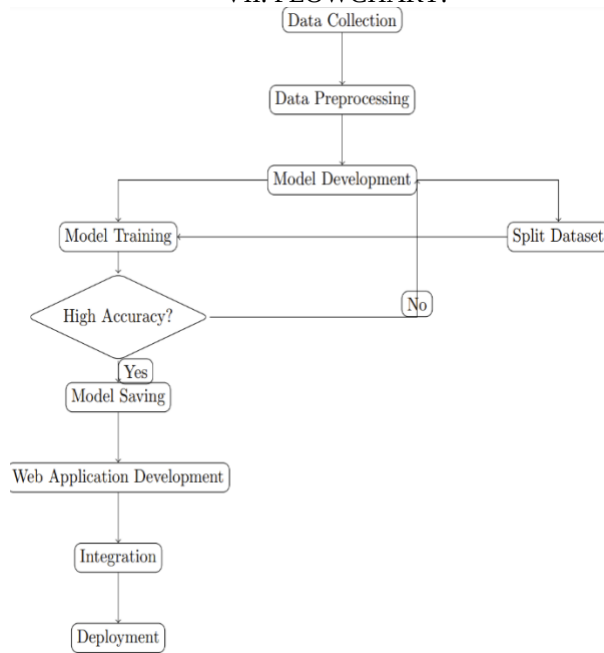


Fig 1.2

VIII. RESULT AND DISCUSSION

The results will be based on the input given. The defected plant image is given as an input and our trained model identifies the disease affected the particular crop precisely. The model achieved an accuracy of 0.9962 with a considerable amount of loss 0.0155.

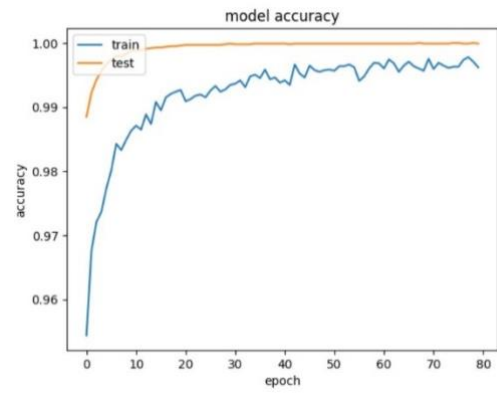


Fig 1.3 Accuracy of model

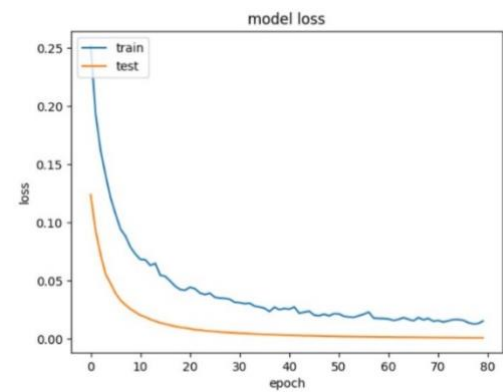


Fig 1.4 Model Loss



(a)



(b)



(a)

Fig 1.5

Fig 1.5 Sample Images of disease affected the apple plant. (a) Black rot , (b) Cedar apple rust.



(a)



(b)

Fig 1.6

Fig 1.6 Sample Images of Disease affected the corn(maize) plant. (a) Common rust, (b) Northern leaf blight.



(b)

Fig 1.7

Fig 1.7 Sample images of disease affected the tomato plant. (a) Tomato mosaic virus, (b) Tomato yellow leaf curl virus

IX. CONCLUSION

The proposed system provides a significant leap forward in the system of plant disease identification through the hybrid deep learning models to the agricultural practices. This systems advantages includes consistent identification and improved accuracy through increased scalability and also a sustainable approach to managing the diseases in plants. It reduces the dependency of manual inspection our system allows access to accurate disease diagnosis benefiting the farmers. This system ensures its adaptability to evolving challenges and emerging diseases for the selected crops. This system has the potential to significantly improve the agricultural productivity by identifying the plant diseases in a more efficient and precise manner for the selected crops.

X. REFERENCE

- [1] Muhammad Hammad Saleem, Johan Potgieter and Khalid Mahmood Arif "Plant Disease Classification: A Comparative Evaluation of Convolutional Neural Networks and Deep Learning Optimizers" 2020 by the authors. Licensee MDPI, Basel, Switzerland. doi:10.3390/plants9101319
- [2] Vani Rajasekar, K. Venu, Soumya Ranjan Jena, R. Janani Varthini and S. Ishwarya. 2021 Detection of Cotton Plant Diseases Using Deep Transfer Learning. Journal of Mobile

- Multimedia, Vol. 18 2, 307–324. doi: 10.13052/jmm1550-4646.1828 River Publishers.
- [3] Ananda S. Paymode , Vandana B. Malode. 2022 Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG. The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co., Ltd. This is an open access article under the CC BY-NC-ND license
- [4] G.Sumalatha, Dr Jhansi Rani Singothu, Dr S. Krishna Rao. Transfer learning-based Plant Disease Detection.
- [5]Muhammad Sufyan Arshad, Usman Abdur Rehman, Muhammad Moazam Fraz. Plant Disease Identification Using Transfer Learning. 2021 International Conference on Digital Futures and Transformative Technologies (ICoDT2) | 978-1-6654-1285-8/20/\$31.00 ©2021 IEEE DOI:10.1109/ICoDT252288.2021.9441512.
- [6] Sakshi Raina,Dr. Abhishek Gupta. A Study on Various Techniques for Plant Leaf Disease Detection Using Leaf Image. 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS) | 978-1-7281-9537-7/20/\$31.00 ©2021IEEE DOI:10.1109/ICAIS50930.2021.9396023.
- [7] Rajbongshi, A., Khan, T., Pramanik, M. M. R. A., Tanvir, S. M., & Siddiquee, N. R. C. (2021). Recognition of mango leaf disease using convolutional neural network models: a transfer learning approach. *Indonesian Journal of Electrical Engineering and Computer Science*, 23(3), 1681-1688.
- [8] Verma, S., Chug, A., Singh, A. P., Sharma, S., & Rajvanshi, P. (2019). Deep learning-based mobile application for plant disease diagnosis: A proof of concept with a case study on tomato plant. In *Applications of image processing and soft computing systems in agriculture* (pp. 242-271). IGI global.
- [9] Aversano, L., Bernardi, M. L., Cimitile, M., Iammarino, M., & Rondinella, S. (2020, November). Tomato diseases classification based on VGG and transfer learning. In *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)* (pp. 129-133). IEEE.
- [10] ÖZDEN, C. (2023). Corn Disease Detection Using Transfer Learning. *Black Sea Journal of Engineering and Science*, 387-393.
- [11] Fauzi, A., Syarif, I., & Badriyah, T. (2023). Development of a Mobile Application for Plant Disease Detection using Parameter Optimization Method in Convolutional Neural Networks Algorithm. *EMITTER International Journal of Engineering Technology*, 11(2), 192-213.
- [12] Trihardianingsih, L., Sunyoto, A., & Hidayat, T. (2023). Classification of Tea Leaf Diseases Based on ResNet-50 and Inception V3. *Sinkron: jurnal dan panellation Teknik informatika*, 8(3), 1564-1573.
- [13] Altuntaş, Y., & KOCAMAZ, F. (2021). Deep feature extraction for detection of tomato plant diseases and pests based on leaf images. *Celal Bayar University Journal of Science*, 17(2), 145-157.
- [14] Ahmed, M. R. (2021). Leveraging convolutional neural network and transfer learning for cotton plant and leaf disease recognition. *Int. J. Image Graph. Signal Process*, 13(4), 47-62.
- [15] Sharma, M., Kumar, C. J., & Deka, A. (2022). Early diagnosis of rice plant disease using machine learning techniques. *Archives of Phytopathology and Plant Protection*, 55(3), 259-283.

D. CONFERENCE CERTIFICATE

